Name: _____

Abiturprüfung 2014

Informatik, Grundkurs

Aufgabenstellung:

In Bussen und U-Bahnen können die Fahrgäste auf modernen Displays erkennen, welches die jeweils nächsten Stationen sind, an denen das Fahrzeug halten wird.

Abbildung 1: Display in einer U-Bahn

Zu jedem Zeitpunkt erkennt der Fahrgast die Namen der drei folgenden Stationen (zum Ende der Fahrt hin ggf. auch weniger als drei) sowie Angaben über die Fahrzeiten.

Wir betrachten jetzt die U-Bahn-Linie 4, die die folgenden Haltestellen benutzt:

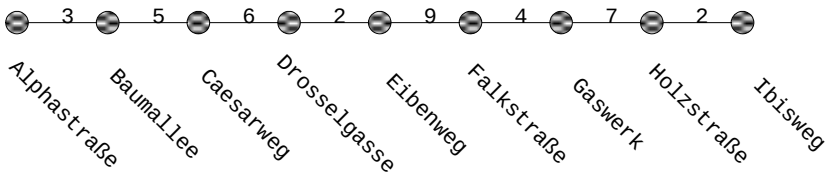


Abbildung 2: Verlauf der U-Bahn-Linie 4

Die Zahlen an den Teilstrecken geben an, wie lange die Bahn für die Strecke zwischen den jeweiligen Haltestellen benötigt (Angabe in Minuten).

Nachdem die Bahn an der „Alphastraße“ in Richtung „Ibisweg“ abgefahren ist, hat das Display in vereinfachter Darstellung das obige Aussehen (Abbildung 1).

Ist die Bahn an der Haltestelle „Ibisweg“ angekommen, fährt sie in der Gegenrichtung bis zur Haltestelle „Alphastraße“; sie hält jetzt an den oben angegebenen Haltestellen in umgekehrter Reihenfolge. Die jeweiligen Fahrzeiten auf den Teilstrecken ändern sich nicht.

Name: _____

a) Die Linie 4 (Abbildung 2) ist auf dem Weg in Richtung „Ibisweg“ gerade am „Caesarweg“ abgefahren.

Stellen Sie in dieser Situation das Display geeignet dar.

Stellen Sie anschließend zwei Displays dar, die ein Fahrgast sieht, nachdem die Bahn auf der Fahrt in Richtung „Ibisweg“ bzw. in Richtung „Alphastraße“ die Haltestelle „Gaswerk“ verlassen hat.

(6 Punkte)

Einige Aspekte der benutzen Software, die die Haltestellen von U-Bahn-Linien verwaltet, sollen hier betrachtet werden.

In einer ersten Version wurden die Klassen `Teilstrecke` und `UBahnLinie` modelliert und implementiert:

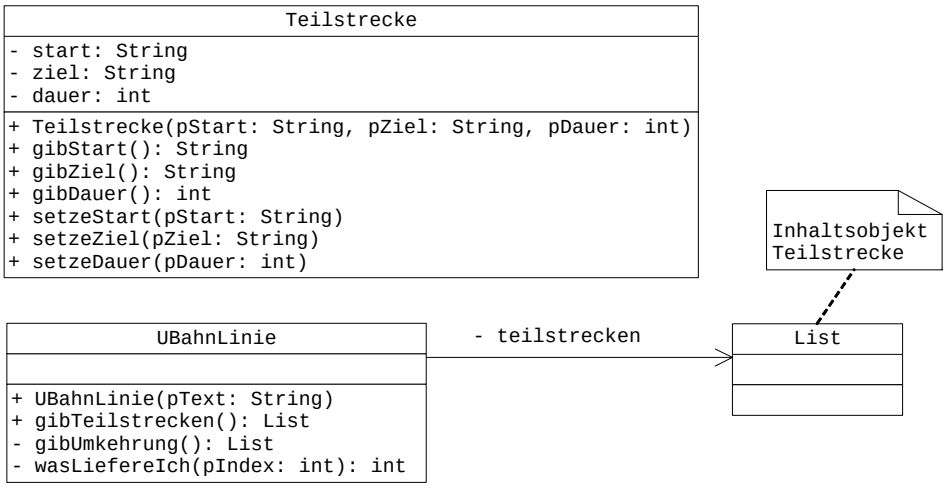


Abbildung 3: Implementationsdiagramm

Die Dokumentationen der Klassen finden Sie im Anhang.

Name: _____

b) In der Klasse `UBahnLinie` existiert unter dem Namen `teilstrecken` ein Attribut der Klasse `List`, das die Teilstrecken (Objekte der Klasse `Teilstrecke`) dieser Linie in der zu durchfahrenden Reihenfolge verwaltet.

Weiterhin gibt es in dieser Klasse eine Methode `wasLiefereIch`:

```
 1 private int wasLiefereIch(int pIndex) {
 2   teilstrecken.toFirst();
 3   int i = 0;
 4   while (teilstrecken.hasAccess() && i < pIndex) {
 5     teilstrecken.next();
 6     i++;
 7   }
 8   int s = 0;
 9   while (teilstrecken.hasAccess()) {
10     Teilstrecke teil =
                      (Teilstrecke) teilstrecken.getObject();
11     s = s + teil.gibDauer();
12     teilstrecken.next();
13   }
14   return s;
15 }
```

Analysieren Sie diese Methode, indem Sie erläutern, wie die Methode für die Linie 4 (Abbildung 2) mit dem Parameterwert `pIndex = 3` *arbeitet.*

Geben Sie für die Linie 4 (Abbildung 2) an, welchen Wert diese Methode beim Aufruf mit dem Parameterwert `pIndex = 3` *liefert.*

Erläutern Sie, welche Funktion die Methode `wasLiefereIch` *im Sachzusammenhang hat.*

(11 Punkte)

Name: _____

c) Ist die Bahn am Ende der jeweiligen Fahrt angekommen, ändert sie ihre Richtung. Die in der Klasse `UBahnLinie` als Attribut benutzte Liste `teilstrecken` der Teilstrecken sowie die darin verwalteten Teilstrecken müssen dazu geeignet erneuert werden.

Implementieren Sie in der Klasse `UBahnLinie` *eine Methode* `gibUmkehrung` *mit dem Methodenkopf*

```
private List gibUmkehrung()
```

die aus der Liste `teilstrecken` *die Liste von Teilstrecken für die Rückfahrt erzeugt.*

(11 Punkte)

d) Um in einer U-Bahn die Anzeige zu verwalten, wurden folgende Modellerweiterungen vorgeschlagen:

- Eine Zeile auf dem Display besteht aus der Angabe einer Haltestelle sowie der zugehörigen Zeitangabe. Dazu soll eine Klasse `DisplayEintrag` entworfen werden.
- Eine Klasse `Display` verwaltet maximal drei Objekte der Klasse `DisplayEintrag`.
- In der Klasse `UBahnLinie` soll es ein Attribut der Klasse `Display` geben, das mit einer Methode `updateDisplay` aktualisiert werden kann:

```
public void updateDisplay(int pAbWo)
```

Dabei gibt der Parameter `pAbWo` an, welche Nummer die Haltestelle hat, die die U-Bahn gerade verlassen hat (die Starthaltestelle hat die Nummer 0).

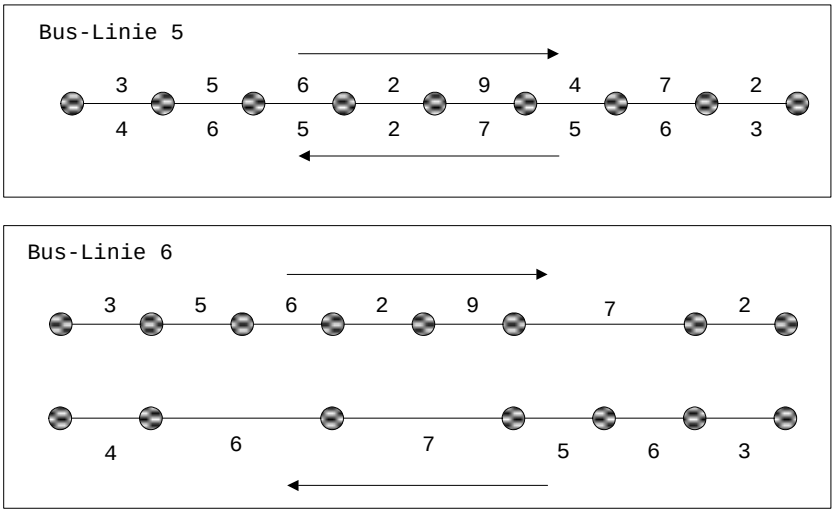
Modellieren Sie die Klassen `DisplayEintrag` *und* `Display`*, indem Sie das Implementationsdiagramm aus Abbildung 3 erweitern. Benutzen Sie dazu das Diagramm in der Anlage (Abbildung 4).*

Dokumentieren Sie die von Ihnen modellierten Methoden und Konstruktoren der Klassen.

(14 Punkte)

Name: _____

e) Das hier für U-Bahn-Linien vorgesehene System kann für viele Buslinien nur bedingt benutzt werden. Folgende Abbildungen zeigen den Haltestellenplan zweier Buslinien (die Namen der Haltestellen sind hier weggelassen):



Begründen Sie, weshalb die für U-Bahn-Linien vorgeschlagene Modellierung (Abbildung 3) für diese Fälle ungeeignet ist.

Entwickeln Sie einen Modellierungsansatz, sodass die beiden oben geschilderten Szenarien abgebildet werden können. Beschreiben Sie umgangssprachlich die von Ihnen geplanten Erweiterungen bzw. Änderungen am bisherigen Modell. Es muss kein neues Implementationsdiagramm angegeben werden.

(8 Punkte)

Zugelassene Hilfsmittel:

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

Name: _____

Anlage

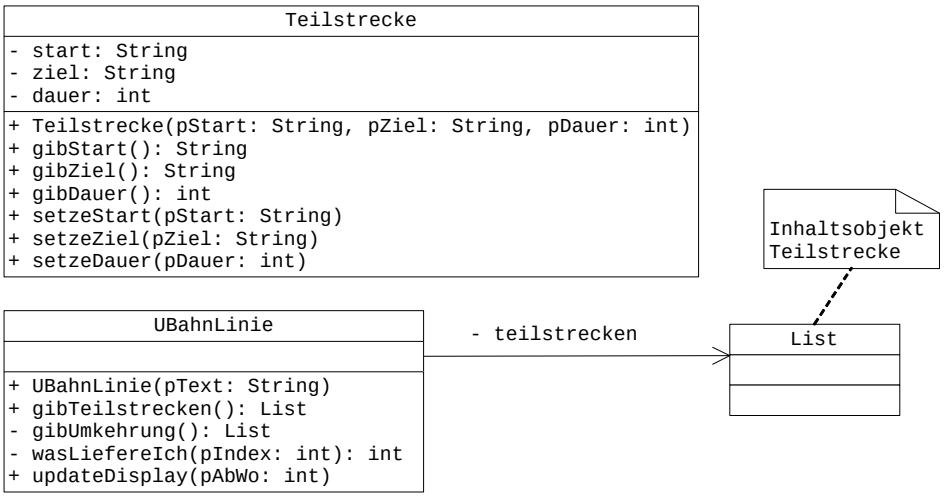


Abbildung 4: Vorlage für das Implementationsdiagramm (Teilaufgabe d))

Name: _____

Anhang

Auszug aus der Dokumentation der Klasse `Teilstrecke`

Ein Objekt dieser Klasse verwaltet Teilstrecken einer U-Bahn-Linie. Es werden die Namen der Start- bzw. Zielhaltestelle sowie die Angabe der Zeit (in Minuten) verwaltet, die die Bahn für diese Teilstrecke benötigt.

Konstruktor `Teilstrecke(String pStart, String pZiel, int pDauer)`
Eine neue Teilstrecke wird erstellt.

Anfrage `String gibStart()`
Die Methode liefert den Namen der Starthaltestelle dieser Teilstrecke.

Anfrage `String gibZiel()`
Die Methode liefert den Namen der Zielhaltestelle dieser Teilstrecke.

Anfrage `int gibDauer()`
Die Methode liefert die Zeit (in Minuten), die die Bahn für diese Teilstrecke benötigt.

Auftrag `void setzeStart(String pStart)`
Der Name der Starthaltestelle wird neu gesetzt.

Auftrag `void setzeZiel(String pZiel)`
Der Name der Zielhaltestelle wird neu gesetzt.

Auftrag `void setzeDauer(int pDauer)`
Die Dauer, die die Bahn für diese Teilstrecke benötigt, wird auf den neuen Wert gesetzt.

Name: _____

Auszug aus der Dokumentation der Klasse `UBahnLinie`

Ein Objekt dieser Klasse verwaltet eine U-Bahn-Linie als Abfolge von Teilstrecken. Dabei ist das Ziel einer jeden Teilstrecke identisch mit dem Start der jeweils folgenden Teilstrecke.

Konstruktor **`UBahnLinie(String pText)`**

Eine neue U-Bahn-Linie wird erstellt. `pText` hat dabei die Form:
`<Name>;<Dauer>;<Name>;<Dauer> … ;<Name>;<Dauer>;<Name>`

Anfrage **`List gibTeilstrecken()`**

Eine Liste von Objekten der Klasse `Teilstrecke` wird geliefert, in der die Teilstrecken dieser U-Bahn-Linie in der aktuell zu durchfahrenden Reihenfolge enthalten sind.

Anfrage **`int wasLiefereIch(int pIndex)`**

Siehe Teilaufgabe b)

Anfrage **`List gibUmkehrung()`**

Siehe Teilaufgabe c)

Auftrag **`void updateDisplay(int pAbWo)`**

Siehe Teilaufgabe d)

Name: _____

Die Klasse `List`

Objekte der Klasse **`List`** verwalten beliebig viele, linear angeordnete Objekte. Auf höchstens ein Listenobjekt, aktuelles Objekt genannt, kann jeweils zugegriffen werden. Wenn eine Liste leer ist, vollständig durchlaufen wurde oder das aktuelle Objekt am Ende der Liste gelöscht wurde, gibt es kein aktuelles Objekt. Das erste oder das letzte Objekt einer Liste können durch einen Auftrag zum aktuellen Objekt gemacht werden. Außerdem kann das dem aktuellen Objekt folgende Listenobjekt zum neuen aktuellen Objekt werden.

Das aktuelle Objekt kann gelesen, verändert oder gelöscht werden. Außerdem kann vor dem aktuellen Objekt ein Listenobjekt eingefügt oder ein Listenobjekt an das Ende der Liste angefügt werden.

Dokumentation der Klasse `List`

Konstruktor **`List()`**

Eine leere Liste wird erzeugt.

Anfrage **`boolean isEmpty()`**

Die Anfrage liefert den Wert `true`, wenn die Liste keine Objekte enthält, sonst liefert sie den Wert `false`.

Anfrage **`boolean hasAccess()`**

Die Anfrage liefert den Wert `true`, wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert `false`.

Auftrag **`void next()`**

Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausführung des Auftrags kein aktuelles Objekt, d. h., `hasAccess()` liefert den Wert `false`.

Auftrag **`void toFirst()`**

Falls die Liste nicht leer ist, wird das erste Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

Auftrag **`void toLast()`**

Falls die Liste nicht leer ist, wird das letzte Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

Name: _____

Anfrage **`Object getObject()`**

Falls es ein aktuelles Objekt gibt (`hasAccess()` == `true`), wird das aktuelle Objekt zurückgegeben, andernfalls (`hasAccess()` == `false`) gibt die Anfrage den Wert `null` zurück.

Auftrag **`void setObject(Object pObject)`**

Falls es ein aktuelles Objekt gibt (`hasAccess()` == `true`) und `pObject` ungleich `null` ist, wird das aktuelle Objekt durch `pObject` ersetzt. Sonst bleibt die Liste unverändert.

Auftrag **`void append(Object pObject)`**

Ein neues Objekt `pObject` wird am Ende der Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Wenn die Liste leer ist, wird das Objekt `pObject` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt (`hasAccess()` == `false`). Falls `pObject` gleich `null` ist, bleibt die Liste unverändert.

Auftrag **`void insert(Object pObject)`**

Falls es ein aktuelles Objekt gibt (`hasAccess()` == `true`), wird ein neues Objekt vor dem aktuellen Objekt in die Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Falls die Liste leer ist und es somit kein aktuelles Objekt gibt (`hasAccess()` == `false`), wird `pObject` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt. Falls es kein aktuelles Objekt gibt (`hasAccess()` == `false`) und die Liste nicht leer ist oder `pObject` gleich `null` ist, bleibt die Liste unverändert.

Auftrag **`void concat(List pList)`**

Die Liste `pList` wird an die Liste angehängt. Anschließend wird `pList` eine leere Liste. Das aktuelle Objekt bleibt unverändert. Falls `pList` `null` oder eine leere Liste ist, bleibt die Liste unverändert.

Auftrag **`void remove()`**

Falls es ein aktuelles Objekt gibt (`hasAccess()` == `true`), wird das aktuelle Objekt gelöscht und das Objekt hinter dem gelöschten Objekt wird zum aktuellen Objekt. Wird das Objekt, das am Ende der Liste steht, gelöscht, gibt es kein aktuelles Objekt mehr (`hasAccess()` == `false`). Wenn die Liste leer ist oder es kein aktuelles Objekt gibt (`hasAccess()` == `false`), bleibt die Liste unverändert.

Name: _____

Die Klasse Stack

Objekte der Klasse **`Stack`** (Keller, Stapel) verwalten beliebige Objekte nach dem Last-In-First-Out-Prinzip, d. h., das zuletzt abgelegte Objekt wird als erstes wieder entnommen.

Dokumentation der Klasse Stack

Konstruktor **`Stack()`**

Ein leerer Stapel wird erzeugt.

Anfrage **`boolean isEmpty()`**

Die Anfrage liefert den Wert `true`, wenn der Stapel keine Objekte enthält, sonst liefert sie den Wert `false`.

Auftrag **`void push(Object pObject)`**

Das Objekt `pObject` wird oben auf den Stapel gelegt. Falls `pObject` gleich `null` ist, bleibt der Stapel unverändert.

Auftrag **`void pop()`**

Das zuletzt eingefügte Objekt wird von dem Stapel entfernt. Falls der Stapel leer ist, bleibt er unverändert.

Anfrage **`Object top()`**

Die Anfrage liefert das oberste Stapelobjekt. Der Stapel bleibt unverändert. Falls der Stapel leer ist, wird `null` zurückgegeben.

Unterlagen für die Lehrkraft

Abiturprüfung 2014

Informatik, Grundkurs

1. Aufgabenart

Aufgabenart	Modellierung einer Problemstellung, Entwurf und Implementation von Algorithmen
Syntaxvariante	Java

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

- entfällt

4. Bezüge zu den Vorgaben 2014

1. Inhaltliche Schwerpunkte
- Datenstrukturen
  - Lineare Strukturen

2. Medien/Materialien
- entfällt

5. Zugelassene Hilfsmittel

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Modelllösungen

Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Das Display könnte die folgende Form haben:

```
Drosselgasse in  6 Min.
Eibenweg     in  8 Min.
Falkstrasse  in 17 Min.
```

Hat die Bahn auf dem Hinweg die Haltestelle „Gaswerk“ verlassen, hat das Display das folgende Aussehen:

```
Holzstrasse  in 7 Min.
Ibisweg      in 9 Min.
```

Auf dem Rückweg, nach Verlassen der Haltestelle „Gaswerk“, hat das Display das folgende Aussehen:

```
Falkstrasse   in  4 Min.
Eibenweg      in 13 Min.
Drosselgasse  in 15 Min.
```

Teilaufgabe b)

Nachdem die erste Schleife die Teilstrecken

(Alphastrasse , Baumallee, 3)
(Baumallee, Caesarweg, 5)
(Caesarweg, Drosselgasse, 6)

durchlaufen hat, bewegt die zweite Schleife den Listenzeiger bis „hinter das Listenende“ und summiert dabei die Teilstreckenlängen in der Variablen `s`, die dabei die Werte `0`, `2`, `11`, `15`, `22`, `24` annimmt.

Die Methode `wasLiefereIch` liefert mit dem Parameter `pIndex = 3` den Wert 24.

Der Wert, den die Methode allgemein liefert, ist die Zeit (in Minuten), die die Bahn benötigt, um von der als Parameter angegebenen Haltestelle (in Form einer laufenden Nummer; die Starthaltestelle hat dabei die Nummer 0) bis zur Endhaltestelle zu kommen.

Teilaufgabe c)

Die Methode `gibUmkehrung` kann mit Hilfe eines Stacks folgendermaßen implementiert werden:

```
private List gibUmkehrung() {
  Stack s = new Stack();
  teilstrecken.toFirst();
  while (teilstrecken.hasAccess()) {
    Teilstrecke teil = (Teilstrecke) teilstrecken.getObject();
    Teilstrecke rueck = new Teilstrecke(teil.gibZiel(),
                                teil.gibStart(), teil.gibDauer());
    s.push(rueck); //umgekehrte Teilstrecke auf den Stack
    teilstrecken.next();
  }  //letzte Teilstrecke liegt oben auf dem Stack
  List rueckfahrt = new List ();
  while (!s.isEmpty()) {
    Teilstrecke teil = (Teilstrecke) s.top();
    s.pop();   //immer hinten anhängen
    rueckfahrt.append(teil);
  }
  return rueckfahrt;
}
```

Alternativ kann ohne einen Stack z. B. wie folgt implementiert werden:

```
private List gibUmkehrung () {
  List rueckfahrt = new List();
  teilstrecken.toFirst();
  while (teilstrecken.hasAccess()) {
    Teilstrecke teil = (Teilstrecke) (teilstrecken.getObject());
    Teilstrecke rueck = new Teilstrecke
               (teil.gibZiel(),teil.gibStart(), teil.gibDauer());
    rueckfahrt.toFirst();
    rueckfahrt.insert(rueck); //immer vorne in die Liste
    teilstrecken.next();
  }
  return rueckfahrt;
}
```

Teilaufgabe d)

Die Klasse `DisplayEintrag` könnte in folgender Form modelliert werden:

Konstruktor **`DisplayEintrag(String pName, int pDauer)`**
Ein neuer Eintrag wird erstellt.

Anfrage **`String gibName()`**
Die Methode liefert den (Haltestellen-)Namen des Eintrags.

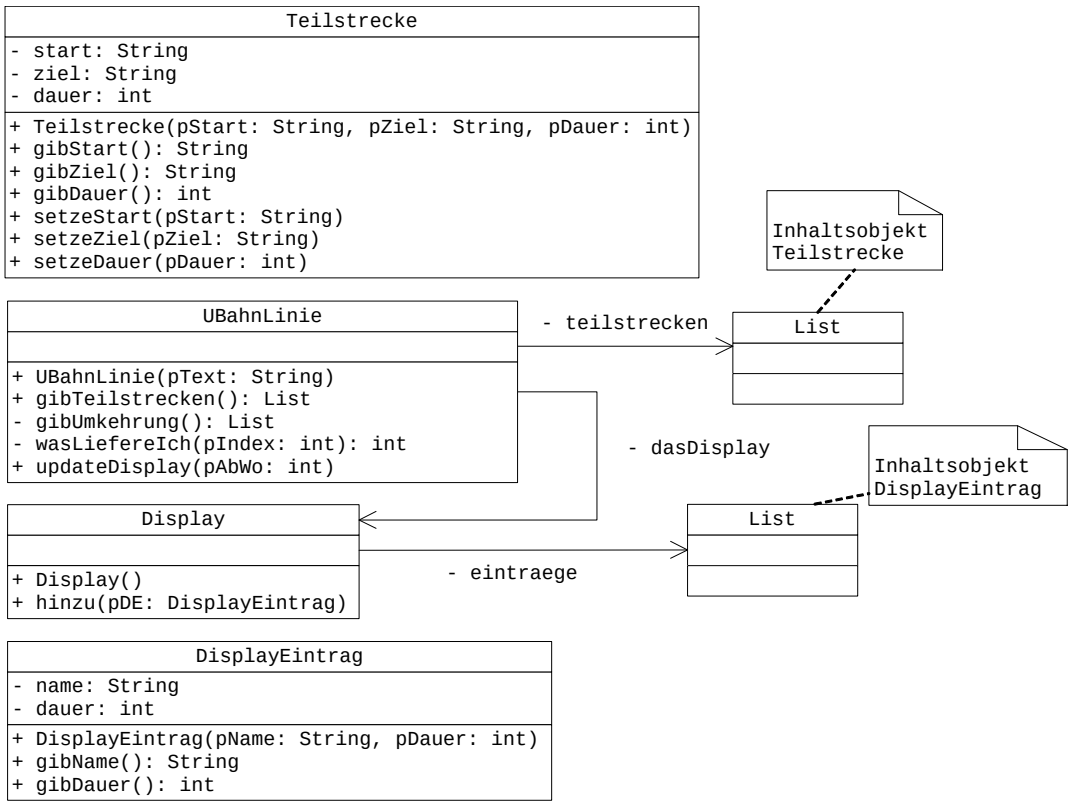
Anfrage **`int gibDauer()`**
Die Methode liefert die Zeit (in Minuten), die der Bus bis zu der Haltestelle benötigt.

Die Klasse `Display` in folgender Form könnte modelliert werden:

Konstruktor **`Display ()`**
Ein neues (leeres) Display wird erstellt.

Auftrag **`void hinzu(DisplayEintrag pDE)`**
Der DisplayEintrag `pDE` wird hinzugefügt, sofern weniger als drei Einträge enthalten sind.

Das Implementationsdiagramm könnte folgendes Aussehen haben:



Korrekt ist auch, wenn das Display die Einträge in einem Array oder in drei Attributen verwaltet.

Teilaufgabe e)

In dem hier für U-Bahn-Linien gewählten Modell sind

a) die Haltestellen auf der Hin- bzw. Rückfahrt grundsätzlich identisch.
b) die Zeiten zwischen je zwei Haltestellen auf dem Hin- und Rückweg identisch.

Die Modellierung ist also nicht für die geschilderten Fälle geeignet.

Mögliche Modelländerungen sind vorstellbar:

- Bei einer Teilstrecke werden zwei (ggf. unterschiedliche) Zeiten (Hinfahrzeit, Rückfahrzeit) als Attribut verwaltet.
- In der Klasse `UBahnLinie` gibt es für die Hin- bzw. Rückfahrt verschiedene Listen von Teilstrecken.
- Es gibt Methoden, die neue Teilstrecken einfügen, vorhandene Teilstrecken löschen und ggf. deren Zeiten ändern.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	stellt das Display in dieser Situation geeignet dar.	2			
2	stellt das Display für die Hinfahrt nach Verlassen der Haltestelle „Gaswerk“ dar.	2			
3	stellt das Display für die Rückfahrt nach Verlassen der Haltestelle „Gaswerk“ dar.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (6) …… ……					
	Summe Teilaufgabe a)	**6**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	gibt die in der ersten Schleife durchlaufenen Teilstrecken an.	3			
2	gibt die Werte der Variablen `s` an.	3			
3	gibt an, welchen Wert die Methode liefert.	2			
4	erläutert die Funktion der Methode im Sachzusammenhang.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (11) …… ……					
	Summe Teilaufgabe b)	**11**			

[2] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	benutzt einen Stack, auf den die umgekehrten Teilstrecken gelegt werden.	6			
2	fügt die Teilstrecken vom Stack in eine neue Liste am Ende ein.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (11) …… ……					
	Summe Teilaufgabe c)	**11**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erweitert das Implementationsdiagramm um die Klasse `DisplayEintrag`.	3			
2	erweitert das Implementationsdiagramm um die Klasse `Display`.	3			
3	erweitert das Implementationsdiagramm der Klasse `UBahnLinie` um ein Attribut vom Typ `Display`.	2			
4	erstellt Dokumentationen für die Methoden der Klasse `DisplayEintrag`.	3			
5	erstellt Dokumentationen für die Methoden der Klasse `Display`.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (14) …… ……					
	Summe Teilaufgabe d)	**14**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	begründet, weshalb die Modellierung für die angegebenen Fälle ungeeignet ist.	2			
2	entwickelt einen Modellierungsansatz.	6			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe e)	**8**			

	Summe insgesamt	**50**			

Festlegung der Gesamtnote (Bitte nur bei der letzten bearbeiteten Aufgabe ausfüllen.)

	Lösungsqualität			
	maximal erreichbare Punktzahl	**EK**	**ZK**	**DK**
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**100**			
aus der Punktsumme resultierende Note				
Note ggf. unter Absenkung um ein bis zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

ggf. arithmetisches Mittel der Punktsummen aus EK und ZK: _____

ggf. arithmetisches Mittel der Notenurteile aus EK und ZK: _____

Die Klausur wird abschließend mit der Note: _____ (____ Punkte) bewertet.

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

Für die Zuordnung der Notenstufen zu den Punktzahlen ist folgende Tabelle zu verwenden:

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	100 – 95
sehr gut	14	94 – 90
sehr gut minus	13	89 – 85
gut plus	12	84 – 80
gut	11	79 – 75
gut minus	10	74 – 70
befriedigend plus	9	69 – 65
befriedigend	8	64 – 60
befriedigend minus	7	59 – 55
ausreichend plus	6	54 – 50
ausreichend	5	49 – 45
ausreichend minus	4	44 – 39
mangelhaft plus	3	38 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0

Name: _____

Abiturprüfung 2014

Informatik, Grundkurs

Aufgabenstellung:

Um Speicherplatz zu sparen oder Übertragungszeiten von Daten zu verkürzen, werden Textdateien komprimiert. Im Folgenden soll eine Komprimierungsmöglichkeit dargestellt, diskutiert und teilweise implementiert werden.

Die Grundidee dieses Komprimierungsverfahrens besteht darin, dass jedes Zeichen aus einem bestimmten Text mit einer möglichst kleinen Anzahl von Bits codiert wird.

Beispiel:
Das Wort `HERBERGE` soll codiert werden. Es kommen die Buchstaben `BEGHR` vor, d. h. insgesamt fünf Zeichen. Für die Codierung des benötigten Zeichensatzes (im Folgenden Alphabet genannt) ist ein 3-Bit-Code erforderlich. Da man bei einem 3-Bit-Code acht Zeichen unterscheiden kann, benötigt man nicht für jedes Zeichen drei Bit. Die ersten vier Zeichen sind hier mit drei Bits codiert, das fünfte Zeichen kann dann mit einem Bit codiert werden:

B	E	G	H	R
000	001	010	011	1

Abbildung 1

Der codierte Text des Wortes HERBERGE sieht wie folgt aus (Binärcode): `01100110000011010001`

Name: _____

Statt in einer Tabelle kann man die Codierung der einzelnen Zeichen auch in einem Baum darstellen.

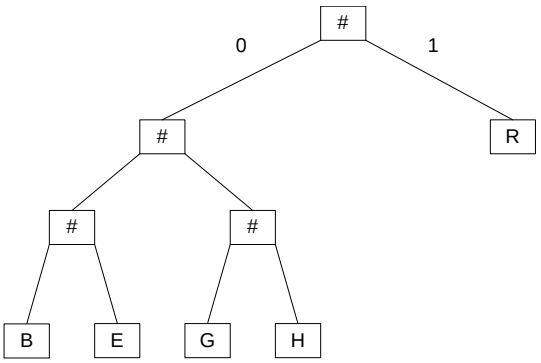


Abbildung 2

In den Blättern des Baumes steht das benötigte Alphabet in sortierter Reihenfolge. In den Ebenen darüber steht ein Sonderzeichen, das in dem Text nicht vorkommt.

Beim Speichern und Übertragen der komprimierten Datei müssen das **sortierte** Alphabet und der Binärcode weitergegeben werden.

a) Gegeben sind die folgende Codetabelle und der Binärcode:

E	G	N	R	T	W
000	001	010	011	10	11

Abbildung 3

Binärcode: `01100000100001011000101000011`

Überführen Sie die Codetabelle in einen Codebaum und bestimmen Sie den Klartext aus dem Binärcode.

(6 Punkte)

Name: _____

b) Gegeben ist der Text `"ERDBEERE"`.

Geben Sie das sortierte Alphabet an und bestimmen Sie den Codebaum und den Binärcode zum Text.

Sowohl das Codieren als auch das Decodieren kann mithilfe der Codetabelle oder mithilfe des Codebaums erfolgen.

Vergleichen Sie die Vorteile bei der Arbeit mit der Codetabelle im Gegensatz zur Arbeit mit dem Codebaum.

(10 Punkte)

Der komprimierte Text besteht aus einem sortierten Alphabet und einem Binärcode. Zur Vereinfachung wird der Binärcode in einem Objekt der Klasse `String` abgelegt. Auf eine effektivere Speicherungsmöglichkeit des Binärcodes wird hier nicht eingegangen.

Grundlage für die folgenden Teilaufgaben ist das Implementationsdiagramm:

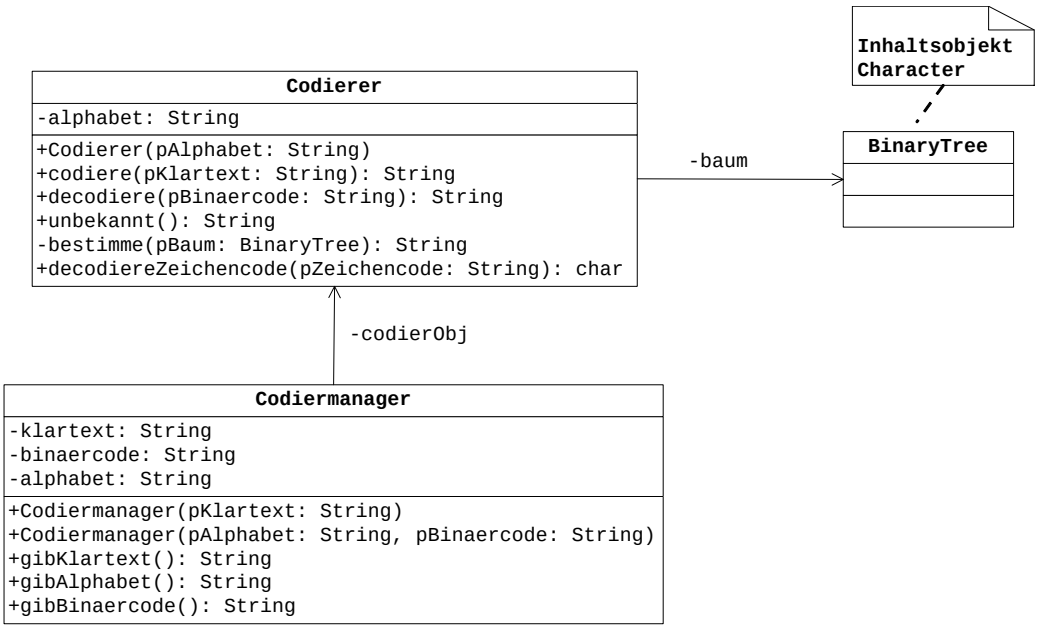


Abbildung 4

Die Dokumentationen der Klassen `Codierer` und `Codiermanager` befinden sich im Anhang.

Name: _____

c) Gegeben sind die Methoden `unbekannt` und `bestimme` der Klasse `Codierer`.

```
 1 public String unbekannt() {
 2   return bestimme(baum);
 3 }

 4 private String bestimme(BinaryTree pBaum) {
 5   if (!pBaum.isEmpty()) {
 6     BinaryTree lBaum = pBaum.getLeftTree();
 7     BinaryTree rBaum = pBaum.getRightTree();
 8     if (lBaum.isEmpty() && rBaum.isEmpty()) {
 9       return "" +
                  ((Character) pBaum.getObject()).charValue();
10     } else {
11       String links = bestimme(lBaum);
12       String rechts = bestimme(rBaum);
13       return links + rechts;
14     }
15   } else {
16     return "";
17   }
18 }
```

Analysieren Sie die Methode `bestimme`*, indem Sie den Ablauf der Methode für den Baum aus Abbildung 2 darstellen. Geben Sie dafür die Belegungen von* `links` *und* `rechts` *(Zeilen 11 und 12) an.*

Erläutern Sie die Funktionalität der Methode.

(12 Punkte)

d) Die Klasse `Codierer` verfügt über eine Methode `decodiereZeichencode`, die aus einem Binärcode für ein Zeichen mithilfe des Codebaums das Zeichen aus dem Alphabet bestimmt.

Entwickeln Sie eine Lösungsidee.

Implementieren Sie die Methode `decodiereZeichencode`.

(14 Punkte)

Name: _____

e) Eine Möglichkeit, Texte zu codieren, besteht darin, dass die Zeichen gemäß der Codierung in der erweiterten ASCII-Tabelle abgespeichert werden. Der Speicherbedarf pro Zeichen beträgt 8 Bit (vgl. Anlage).

Analysieren Sie das dargestellte Komprimierungsverfahren und die Codierung mithilfe der ASCII-Tabelle und ermitteln Sie jeweils Vorzüge der Verfahren.

Beurteilen Sie das dargestellte Komprimierungsverfahren im Vergleich zur Codierung mithilfe der erweiterten ASCII-Tabelle.

(8 Punkte)

Zugelassene Hilfsmittel:

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

Name: _____

Anlage

ASCII ist eine Abkürzung für „American Standard Code for Information Interchange“. Dieser Code wird benutzt, um Zeichen, die gespeichert oder übertragen werden sollen, zu codieren.

Auszug aus einer ASCII-Code-Tabelle im 8-Bit-Format.

Dez	Bin	Zeichen
65	01000001	A
66	01000010	B
…	…	…

Abbildung 5

Name: _____

Anhang

Dokumentation der Klasse `Codierer`

Konstruktor `Codierer(String pAlphabet)`

Zu einem gegebenen Alphabet wird für jedes Zeichen, das im Alphabet vorkommt, ein Binärcode festgelegt.

Anfrage `String codiere(String pKlartext)`

Zu einem gegebenen Klartext wird der Binärcode zum Text bestimmt. Grundlage ist die im Konstruktor gebildete Zuordnung zwischen Zeichen und Binärcode.

Anfrage `String decodiere(String pBinaercode)`

Zu einem gegebenen Binärcode wird der Klartext bestimmt. Grundlage ist die im Konstruktor gebildete Zuordnung zwischen Zeichen und Binärcode.

Anfrage `String unbekannt()`

vergleiche Teilaufgabe c).

Anfrage `String bestimme(BinaryTree pBaum)`

vergleiche Teilaufgabe c).

Anfrage `char decodiereZeichencode(String pZeichenCode)`

vergleiche Teilaufgabe d).

Name: _____

Auszug aus der Dokumentation der Klasse Codiermanager

Konstruktor **`Codiermanager(String pKlartext)`**

Ein Objekt der Klasse `Codiermanager` wird erzeugt. Aus dem gegebenen Klartext werden alle vorkommenden Zeichen bestimmt.

Konstruktor **`Codiermanager(String pAlphabet, String pBinaercode)`**

Ein Objekt der Klasse `Codiermanager` wird erzeugt. Der Parameter `pAlphabet` enthält alle Zeichen, der Parameter `pBinaercode` enthält den Binärcode zu einem Text.

Anfrage **`String gibKlartext()`**

liefert den Klartext.

Anfrage **`String gibAlphabet()`**

liefert das Alphabet.

Anfrage **`String gibBinaercode()`**

liefert den Binärcode.

Name: _____

Die Klasse BinaryTree

Mithilfe der Klasse **`BinaryTree`** können beliebig viele Inhaltsobjekte in einem Binärbaum verwaltet werden. Ein Objekt der Klasse stellt entweder einen leeren Baum dar oder verwaltet ein Inhaltsobjekt sowie einen linken und einen rechten Teilbaum, die ebenfalls Objekte der Klasse `BinaryTree` sind.

Dokumentation der Klasse BinaryTree

Konstruktor **`BinaryTree()`**

Nach dem Aufruf des Konstruktors existiert ein leerer Binärbaum.

Konstruktor **`BinaryTree(Object pObject)`**

Wenn der Parameter `pObject` ungleich `null` ist, existiert nach dem Aufruf des Konstruktors der Binärbaum und hat `pObject` als Inhaltsobjekt und zwei leere Teilbäume. Falls der Parameter `null` ist, wird ein leerer Binärbaum erzeugt.

Konstruktor **`BinaryTree(Object pObject, BinaryTree pLeftTree, BinaryTree pRightTree)`**

Wenn der Parameter `pObject` ungleich `null` ist, wird ein Binärbaum mit `pObject` als Inhaltsobjekt und den beiden Teilbäume `pLeftTree` und `pRightTree` erzeugt. Sind `pLeftTree` oder `pRightTree` gleich `null`, wird der entsprechende Teilbaum als leerer Binärbaum eingefügt. Wenn der Parameter `pObject` gleich `null` ist, wird ein leerer Binärbaum erzeugt.

Anfrage **`boolean isEmpty()`**

Diese Anfrage liefert den Wahrheitswert `true`, wenn der Binärbaum leer ist, sonst liefert sie den Wert `false`.

Auftrag **`void setObject(Object pObject)`**

Wenn der Binärbaum leer ist, wird der Parameter `pObject` als Inhaltsobjekt sowie ein leerer linker und rechter Teilbaum eingefügt. Ist der Binärbaum nicht leer, wird das Inhaltsobjekt durch `pObject` ersetzt. Die Teilbäume werden nicht geändert. Wenn `pObject` `null` ist, bleibt der Binärbaum unverändert.

Anfrage **`Object getObject()`**

Diese Anfrage liefert das Inhaltsobjekt des Binärbaums. Wenn der Binärbaum leer ist, wird `null` zurückgegeben.

Name: _____

Auftrag **`void setLeftTree(BinaryTree pTree)`**

Wenn der Binärbaum leer ist, wird `pTree` nicht angehängt. Andernfalls erhält der Binärbaum den übergebenen Baum als linken Teilbaum. Falls der Parameter `null` ist, ändert sich nichts.

Auftrag **`void setRightTree(BinaryTree pTree)`**

Wenn der Binärbaum leer ist, wird `pTree` nicht angehängt. Andernfalls erhält der Binärbaum den übergebenen Baum als rechten Teilbaum. Falls der Parameter `null` ist, ändert sich nichts.

Anfrage **`BinaryTree getLeftTree()`**

Diese Anfrage liefert den linken Teilbaum des Binärbaumes. Der Binärbaum ändert sich nicht. Wenn der Binärbaum leer ist, wird `null` zurückgegeben.

Anfrage **`BinaryTree getRightTree()`**

Diese Anfrage liefert den rechten Teilbaum des Binärbaumes. Der Binärbaum ändert sich nicht. Wenn der Binärbaum leer ist, wird `null` zurückgegeben.

Unterlagen für die Lehrkraft

Abiturprüfung 2014

Informatik, Grundkurs

1. Aufgabenart

Aufgabenart	Modellierung einer Problemstellung, Entwurf und Implementation von Algorithmen
Syntaxvariante	Java

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

- entfällt

4. Bezüge zu den Vorgaben 2014

1. *Inhaltliche Schwerpunkte*
   Datenstrukturen
   - Baumstrukturen mit den Akzenten
     - Binärbaum
       Anwendung der Standardoperationen
       Traversierungsalgorithmen

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

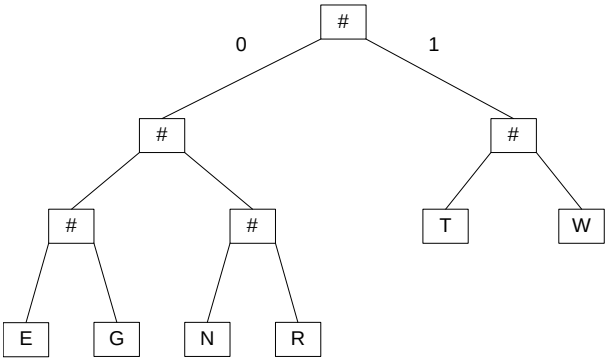
[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Modelllösungen

Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Baumdarstellung:



Binärcode: `011 000 001 000 010 11 000 10 10 000 011`

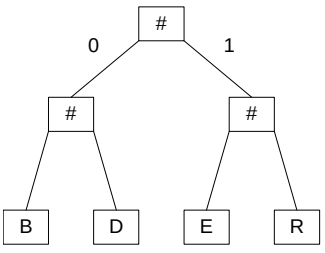
Klartext: `R E G E N W E T T E R`

Teilaufgabe b)

Zur Darstellung des Alphabets reicht ein 2-Bit-Code.

Alphabet: `BDER`

Baumdarstellung:



Klartext: `E R D B E E R E`

Binärcode: `10 11 01 00 10 10 11 10`

Beim Codieren ist die Arbeit mit der Tabelle vorteilhaft. Im geordneten Alphabet kann man schnell das Zeichen finden, die Strategie des binären Suchens ist hier anwendbar.
Beim Decodieren ist die Baumdarstellung vorteilhaft. Die Anzahl der Suchschritte entspricht der Codelänge.

Teilaufgabe c)

links	rechts
B	E
G	H
BE	GH
BEGH	R

Es erfolgt ein vollständiger Baumdurchlauf. Sobald ein Blatt erreicht wird, wird das Zeichen, das im Blatt gespeichert wird, zurückgegeben.

Die Methode liefert das Alphabet als String zurück.

Teilaufgabe d)

Der Binärcode wird als Parameter übergeben. Er wird „bitweise“ abgearbeitet. Falls eine 0 vorliegt, wird der linke Teilbaum des aktuellen Teilbaums gewählt, andernfalls der rechte Teilbaum. Falls ein Blatt vorliegt, enthält das Inhaltsobjekt das gesuchte Zeichen, es wird zurückgegeben.

```
public char decodiereZeichencode(String pZeichenCode) {
  char zeichen = '?';
  BinaryTree teilbaum = baum;
  for (int zaehler = 0; zaehler < pZeichenCode.length();
                                                  zaehler ++) {
    if (pZeichenCode.charAt(zaehler) == '0') {
      teilbaum = teilbaum.getLeftTree();
    } else {
      teilbaum = teilbaum.getRightTree();
    }
  }
  if (teilbaum.getLeftTree().isEmpty()
      && teilbaum.getRightTree().isEmpty() ) {
    // Blatt ist erreicht
    zeichen = ((Character) teilbaum.getObject()).charValue();
  }
  return zeichen;
}
```

Teilaufgabe e)

Verfahren 1: Codierung mithilfe der erweiterten ASCII-Tabelle.

Ein Vorteil besteht darin, dass die Tabelle nicht mit transportiert werden muss. Die ASCII-Tabelle ist ein Standard, auf den zurückgegriffen werden kann. Jedes Zeichen des Textes hat eine binäre Codierung der Länge 8.

Verfahren 2: Codierung mithilfe des dargestellten Verfahrens.

Jedes Zeichen wird mit einer individuellen Codelänge abgespeichert. Die Codelänge ist abhängig von der Anzahl der verwendeten Zeichen (Zeichen im Alphabet). Der Nachteil ist, dass jeder Text eine unterschiedliche Menge von verwendeten Zeichen hat. Das textabhängige Alphabet muss mit abgespeichert oder übermittelt werden.

Das hier dargestellte zweite Verfahren führt bei langen Texten, die nur wenige unterschiedliche Zeichen verwenden, zu einem geringeren Speicherbedarf als die Codierung mit der ASCII-Tabelle.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	überführt die Codetabelle in einen Codebaum.	3			
2	bestimmt den Klartext.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (6) …… ……					
	Summe Teilaufgabe a)	**6**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	gibt das Alphabet an.	2			
2	bestimmt den Codebaum.	2			
3	bestimmt den Binärcode.	3			
4	vergleicht die Arbeit mit Codetabelle und Codebaum beim Encodieren und Decodieren.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe b)	**10**			

[2] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	bestimmt die Belegung der Variablen `links` und `rechts`.	4			
2	stellt den Ablauf der Methode geeignet dar.	4			
3	erläutert die Funktionalität der Methode.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe c)	**12**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt die Lösungsidee.	6			
2	implementiert die Methode `decodiereZeichencode`.	8			
Sachlich richtige Lösungsalternative zur Modelllösung: (14) …… ……					
	Summe Teilaufgabe d)	**14**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert die Codierung mithilfe der erweiterten ASCII-Tabelle und ermittelt die Vorzüge.	3			
2	analysiert das beschriebene Codier-Verfahren und ermittelt die Vorzüge.	3			
3	bewertet die Verfahren im Hinblick auf den Speicherbedarf.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe e)	**8**			

	Summe insgesamt	**50**			

Festlegung der Gesamtnote (Bitte nur bei der letzten bearbeiteten Aufgabe ausfüllen.)

	Lösungsqualität			
	maximal erreichbare Punktzahl	**EK**	**ZK**	**DK**
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**100**			
aus der Punktsumme resultierende Note				
Note ggf. unter Absenkung um ein bis zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

ggf. arithmetisches Mittel der Punktsummen aus EK und ZK: _____

ggf. arithmetisches Mittel der Notenurteile aus EK und ZK: _____

Die Klausur wird abschließend mit der Note: _____ (____ Punkte) bewertet.

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

Für die Zuordnung der Notenstufen zu den Punktzahlen ist folgende Tabelle zu verwenden:

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	100 – 95
sehr gut	14	94 – 90
sehr gut minus	13	89 – 85
gut plus	12	84 – 80
gut	11	79 – 75
gut minus	10	74 – 70
befriedigend plus	9	69 – 65
befriedigend	8	64 – 60
befriedigend minus	7	59 – 55
ausreichend plus	6	54 – 50
ausreichend	5	49 – 45
ausreichend minus	4	44 – 39
mangelhaft plus	3	38 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0

Name: _____

Abiturprüfung 2014

Informatik, Grundkurs

Aufgabenstellung:

Moderne Autos verfügen über Sensoren, mit denen während der Fahrt wichtige Motorfunktionen überprüft werden. Sie werden zur Optimierung der Motorleistung, zur Minimierung des Verbrauchs und zur Fehlerdiagnose verwendet.

Im Folgenden wird von einer einfachen Motorüberwachung ausgegangen. Sobald der Motor des Autos gestartet ist, werden in kurzen Abständen Motordiagnosen (Messungen) durchgeführt, die zu folgenden Ergebnissen führen können:

o	:	**O**ptimale Motorfunktion: Der Motor läuft entsprechend seiner Spezifikation.
l	:	**L**eichter Fehler: Der Motor läuft nicht optimal. Das kann an einem kurzzeitigen, tolerierbaren Fehler oder an einem echten Motorschaden liegen.
k	:	**K**ritischer Fehler: Der Motor läuft nicht korrekt.

Der folgende Zustandsübergangsgraph gehört zu einem deterministischen, endlichen Automaten, der eine Analyse mehrerer Motordiagnosen durchführt. Geht der Automat in einen Endzustand, so wird der Fahrer aufgefordert, eine Werkstatt aufzusuchen, indem am Armaturenbrett die Motorwarnlampe angeht.

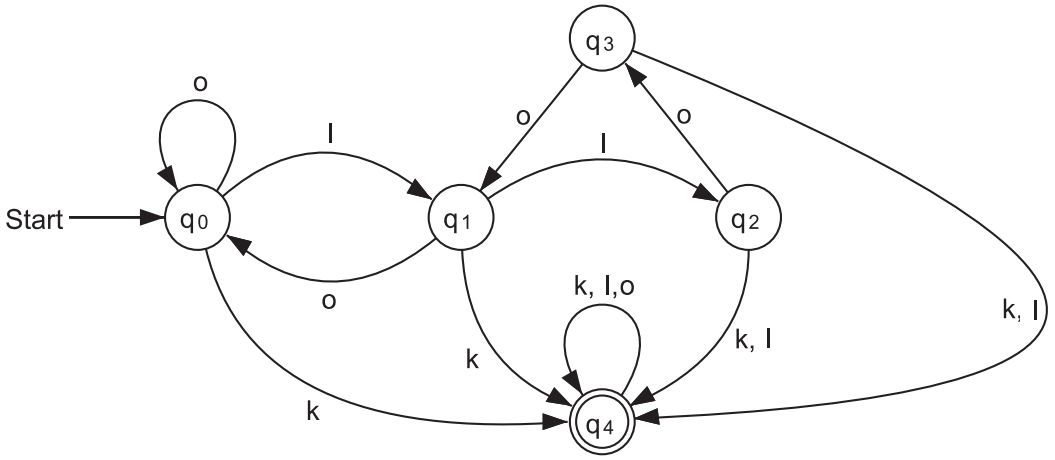


Abbildung 1: Zustandsübergangsgraph zur Motorwarnlampe

Name: _____

a) *Geben Sie zwei verschiedene Worte an, die vom Automaten zum Übergangsgraphen in Abbildung 1 akzeptiert werden und mehr als vier Eingabezeichen haben.*

Bestimmen Sie die Zustandsfolge desselben Automaten bei der Eingabe des Wortes `llooollo` *und geben Sie an, ob das Wort akzeptiert wird.*

Erläutern Sie auf Grundlage des Übergangsgraphen in Abbildung 1, unter welchen Umständen der Fahrer zum Werkstattbesuch aufgefordert wird.

(10 Punkte)

b) Der Automat zum Übergangsgraphen in Abbildung 1 soll als Teil einer Motorsimulation in einem Rechner gespeichert werden. Statt des Zustandsübergangsgraphen soll eine Zustandsübergangstabelle zum Einsatz kommen.
Entwickeln Sie die Zustandsübergangstabelle zum Übergangsgraphen aus Abbildung 1.

Erläutern Sie den Aufbau der Zustandsübergangstabelle.

(10 Punkte)

c) Im Vergleich zu PKWs haben LKWs eine besonders hohe Kilometerlaufleistung und sind besonderen Belastungen ausgesetzt. Die Motoren sind entsprechend konzipiert und sollen über ein anderes Motorwarnsystem als die PKWs verfügen.

Das System soll folgende Eigenschaften aufweisen:

- Zwei Messungen mit dem Ergebnis *kritischer Fehler* (k) sollen zu einer Warnmeldung führen, wenn sie direkt nacheinander auftreten.
- Drei Messungen, die in direkter Folge das Ergebnis *leichter Fehler* (l) oder *kritischer Fehler* (k) haben, sollen zu einer Warnmeldung führen.
- Wird eine Warnmeldung ausgegeben, kann keine zukünftige Messung sie mehr zurücknehmen. Andere Messungen mit dem Ergebnis *optimale Motorfunktion* (o) setzen das System in den Startzustand zurück.

Modellieren Sie einen deterministischen, endlichen Automaten mit den obigen Eigenschaften, indem Sie die notwendigen Mengen und den Zustandsübergangsgraphen angeben.

(18 Punkte)

Name: _____

d) Die folgende Grammatik stellt einen neuen Vorschlag zur Überprüfung der Motorfunktion dar. Die Warnlampe soll immer dann leuchten, wenn das aus den Diagnosemessungen zusammengesetzte Wort zur Sprache dieser Grammatik gehört.

```
Terminale:       {o, l, k}

Nicht-Terminale: {S, A, F}

Startsymbol:     S

Produktionen: {

    S → AFA| FA | AF | F
    A → oA | lA | kA | o | l | k
    F → ll | k

}
```

Begründen Sie, warum die obige Grammatik nicht regulär ist.

Erläutern Sie im Kontext, welche Eigenschaften die Worte aufweisen, die sich mithilfe dieser Grammatik ableiten lassen. Erläutern Sie dazu zunächst, welche Zeichenketten sich allein aus dem Nicht-Terminal A und aus dem Nicht-Terminal F ableiten lassen.

(12 Punkte)

Zugelassene Hilfsmittel:

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

Unterlagen für die Lehrkraft

Abiturprüfung 2014

Informatik, Grundkurs

1. Aufgabenart

Aufgabenart	Aufgabenstellung aus dem Bereich endliche Automaten und formale Sprachen

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

- entfällt

4. Bezüge zu den Vorgaben 2014

1. Inhaltliche Schwerpunkte
Endliche Automaten und formale Sprachen
- Modellieren kontextbezogener Problemstellungen als deterministische endliche Automaten
- Darstellung von deterministischen endlichen Automaten als Graph und als Tabelle
- Formale Sprachen: Reguläre Sprachen und ihre Grammatiken

2. Medien/Materialien
- entfällt

5. Zugelassene Hilfsmittel

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Modelllösungen

Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Folgende Worte werden vom Automaten zum Übergangsgraphen in Abbildung 1 akzeptiert:

1. `lloook`
2. `lolololll`

Die Zustandsfolge zum Wort `llooollo` ist die folgende:

`q0 → q1 → q2 → q3 → q1 → q0 → q1 → q2 → q3`

Das Wort `llooollo` wird nicht vom Automaten akzeptiert, da er nach seiner Abarbeitung in Zustand `q3` ist und es sich dabei nicht um einen Endzustand des Automaten handelt.

Wird die Motorwarnlampe mit einem Automaten zum Übergangsgraphen aus Abbildung 1 gesteuert, so wird der Fahrer unter folgenden Umständen zum Werkstattbesuch aufgerufen:

Die Motorwarnlampe geht immer an, wenn ein *kritischer Fehler* (k) festgestellt wird, da der Automat unabhängig von seinem aktuellen Zustand mit diesem Eingabezeichen sofort in den Endzustand `q4` geht. Ist die Warnlampe einmal an, geht sie auch nicht wieder aus, da alle zukünftigen Messungen, unabhängig von ihrem Ergebnis, den Automaten im Endzustand `q4` belassen.

Leichte Fehler (l) führen nur dann in den Endzustand `q4` und zu einem Angehen der Motorwarnlampe, wenn sie mehrfach auftreten und nicht durch Messungen, die eine *optimale Motorfunktion* (o) dokumentieren, ausgeglichen werden.

Ein *leichter Fehler* (l) kann durch eine Messung einer *optimalen Motorfunktion* (o) ausgeglichen werden. Wird er nicht ausgeglichen, muss ein zweiter *leichter Fehler* (l) durch die Messung von gleich zwei *optimalen Motorfunktionen* (o) ausgeglichen werden.

Um zwei *leichte Fehler* (l) in Folge vollständig auszugleichen, bedarf es also insgesamt dreier fehlerfreier Messungen (o).

Werden zwei *leichte Fehler* (l) nicht ausgeglichen, führt der dritte *leichte Fehler* (l) in den Endzustand `q4` und die Warnlampe wird eingeschaltet.

Teilaufgabe b)

Die Zustandsübergangstabelle zum Automaten des Übergangsgraphen aus Abbildung 1 ist die folgende:

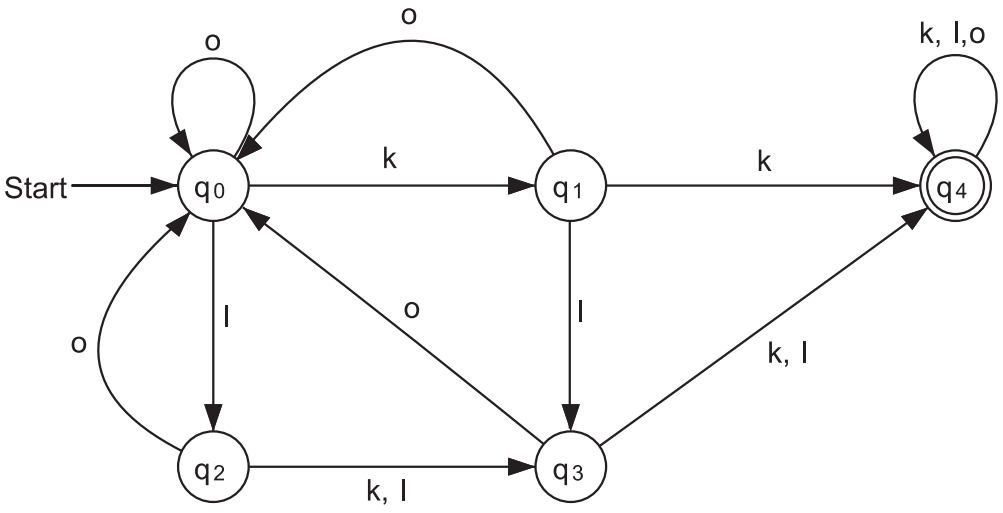
Zustand	**Zustandsübergang**		
	o	**l**	**k**
q0	q0	q1	q4
q1	q0	q2	q4
q2	q3	q4	q4
q3	q1	q4	q4
q4	q4	q4	q4

Die Tabelle beinhaltet die Information, in welchem Zustand mit welchem Zeichen in welchen Zustand gewechselt wird. Die Tabelle besteht aus vier Spalten. In der ersten Spalte sind alle Zustände eingetragen. In den Spalten danach sind die Zustände eingetragen, in die mit dem jeweiligen Zeichen, das die Spalte überschreibt, gewechselt wird.

Teilaufgabe c)

Der folgende Automat entspricht den Anforderungen der Aufgabenstellung:

```
Eingabezeichen: {o, l, k}
Zustände: {q0, q1, q2, q3, q4}
Startzustand: q0
Endzustände: {q4}
Zustandsübergangsgraph:
```



Teilaufgabe d)

Die Grammatik ist nicht regulär, da nicht alle Produktionen den Kriterien einer regulären Grammatik entsprechen. *Beispiel*: S → AFA

Aus dem Nicht-Terminal A lassen sich alle Zeichenketten ableiten, die aus den Zeichen o, l und k bestehen.

Aus dem Nicht-Terminal F lassen sich nur die Zeichenkette ll und das Zeichen k ableiten.

Das bedeutet, dass sich aus dem Startsymbol S Zeichenketten ableiten lassen, die aus einer beliebigen Folge der Zeichen o, l und k bestehen und mindestens einmal die Zeichenketten ll oder das Zeichen k enthalten. Die Zeichenkette ll und das Zeichen k gehören allein ebenfalls zur Sprache.

Im Kontext bedeutet das Folgendes:

Die Warnleuchte wird immer dann angeschaltet, d. h., das entsprechende Wort gehört zur Sprache dieser Grammatik, wenn eine oder mehrere Messungen einen *kritischen Fehler* (k) ergeben haben.

Darüber hinaus geht die Warnleuchte an, wenn zwei Messungen, die einen *leichten Fehler* (l) ergeben haben, direkt aufeinander folgen.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	gibt zwei Worte an, die vom Automaten akzeptiert werden.	2			
2	bestimmt die Zustandsfolge bei der Eingabe des Wortes und gibt an, ob das Wort akzeptiert wird.	4			
3	erläutert, wann der Fahrer zum Werkstattbesuch aufgefordert wird.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe a)	**10**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt die Zustandsübergangstabelle.	5			
2	erläutert den Aufbau der Zustandsübergangstabelle.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe b)	**10**			

[2] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	gibt die Menge der Eingabezeichen an.	2			
2	gibt die Menge der Zustände an.	2			
3	gibt den Startzustand an.	2			
4	gibt die Menge der Endzustände an.	2			
5	gibt den Zustandsübergangsgraphen an.	10			
Sachlich richtige Lösungsalternative zur Modelllösung: (18) …… ……					
	Summe Teilaufgabe c)	**18**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	begründet, warum die Grammatik nicht regulär ist.	4			
2	erläutert, welche Zeichenketten sich aus dem Nicht-Terminal A ergeben.	2			
3	erläutert, welche Zeichenketten sich aus dem Nicht-Terminal F ergeben.	2			
4	analysiert die Eigenschaften der Worte im Kontext der Aufgabenstellung.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe d)	**12**			

	Summe insgesamt	**50**			

Festlegung der Gesamtnote (Bitte nur bei der letzten bearbeiteten Aufgabe ausfüllen.)

	Lösungsqualität			
	maximal erreichbare Punktzahl	EK	ZK	DK
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**100**			
aus der Punktsumme resultierende Note				
Note ggf. unter Absenkung um ein bis zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

ggf. arithmetisches Mittel der Punktsummen aus EK und ZK: _____

ggf. arithmetisches Mittel der Notenurteile aus EK und ZK: _____

Die Klausur wird abschließend mit der Note: _____ (____ Punkte) bewertet.

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

Für die Zuordnung der Notenstufen zu den Punktzahlen ist folgende Tabelle zu verwenden:

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	100 – 95
sehr gut	14	94 – 90
sehr gut minus	13	89 – 85
gut plus	12	84 – 80
gut	11	79 – 75
gut minus	10	74 – 70
befriedigend plus	9	69 – 65
befriedigend	8	64 – 60
befriedigend minus	7	59 – 55
ausreichend plus	6	54 – 50
ausreichend	5	49 – 45
ausreichend minus	4	44 – 39
mangelhaft plus	3	38 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0

Name: _____

Abiturprüfung 2014

Informatik, Grundkurs

Aufgabenstellung:

Ein Bioladen möchte seinen Kunden individuelle Marmeladenmischungen anbieten. Um unnötige Wartezeiten wegen der Zubereitung der Marmeladen zu vermeiden, sollen die Kunden schon von zu Hause aus ihre Wunschmarmelade online zusammenstellen können.

Dazu meldet sich ein Kunde mit seinem Computer beim Server des Bioladens an und kann dann direkt mit der Bestellung seiner Wunschmarmelade anfangen. Die Kommunikation läuft dabei auf Grundlage eines Protokolls ab, das im Folgenden beispielhaft für den Bestellvorgang einer 500-g-Erdbeer-Bananen-Marmelade angegeben ist.

Client sendet an Server	Server sendet an Client
stellt eine Verbindung zum Server her...	`+OK Beginnen Sie mit der Mischung. Verwenden Sie die Befehle ERGAENZE, ENTFERNE, MENGE, VORSCHAU und FERTIG.`
`ERGAENZE Erdbeere`	`+OK Fruchtsorte Erdbeere hinzugefügt.`
`ERGAENZE Banane`	`+OK Fruchtsorte Banane hinzugefügt.`
`ERGAENZE Erdbeere`	`+OK Fruchtsorte Erdbeere hinzugefügt.`
`MENGE 500`	`+OK Menge 500 Gramm geändert.`

`VORSCHAU`	`+OK Zutaten: Erdbeere:Banane:Erdbeere`
	`+OK Menge:500 Gramm`
	`+OK Preis:3,60 Euro`

`ENTFERNE Erdbeere`	`+OK Fruchtsorte Erdbeere gelöscht.`

`VORSCHAU`	`+OK Zutaten: Banane:Erdbeere`
	`+OK Menge:500 Gramm`
	`+OK Preis:3,70 Euro`

`FERTIG`	`+OK Sie können Ihre Wunschmarmelade ab morgen in unserem Bioladen abholen.` ... und trennt die Verbindung.

Abbildung 1: Kommunikationsprotokoll für einen beispielhaften Bestellvorgang

Der Bioladen bietet lediglich die Fruchtsorten Erdbeere, Banane, Kirsche und Apfel an. Außerdem können nur die Mengen 200 g, 300 g und 500 g bestellt werden.

Name: _____

a) Die oben dargestellte Kommunikation verlief ohne Probleme, allerdings ist das nicht immer sichergestellt.

Erläutern Sie zwei Situationen, in denen der Server mit einer Fehlermeldung reagieren müsste.

Geben Sie ein allgemeines Protokoll für die Kommunikation zwischen Client und Server an, welches auch die zuvor genannten Fehleingaben seitens des Clients berücksichtigt.

Geben Sie an, zu welcher Schicht des TCP/IP-Referenzmodells dieses Protokoll gehört.

(9 Punkte)

Das folgende Diagramm stellt einen Ausschnitt des Implementationsdiagramms der Serveranwendung dar. Einen Auszug der Dokumentation dieser Klassen finden Sie im Anhang.

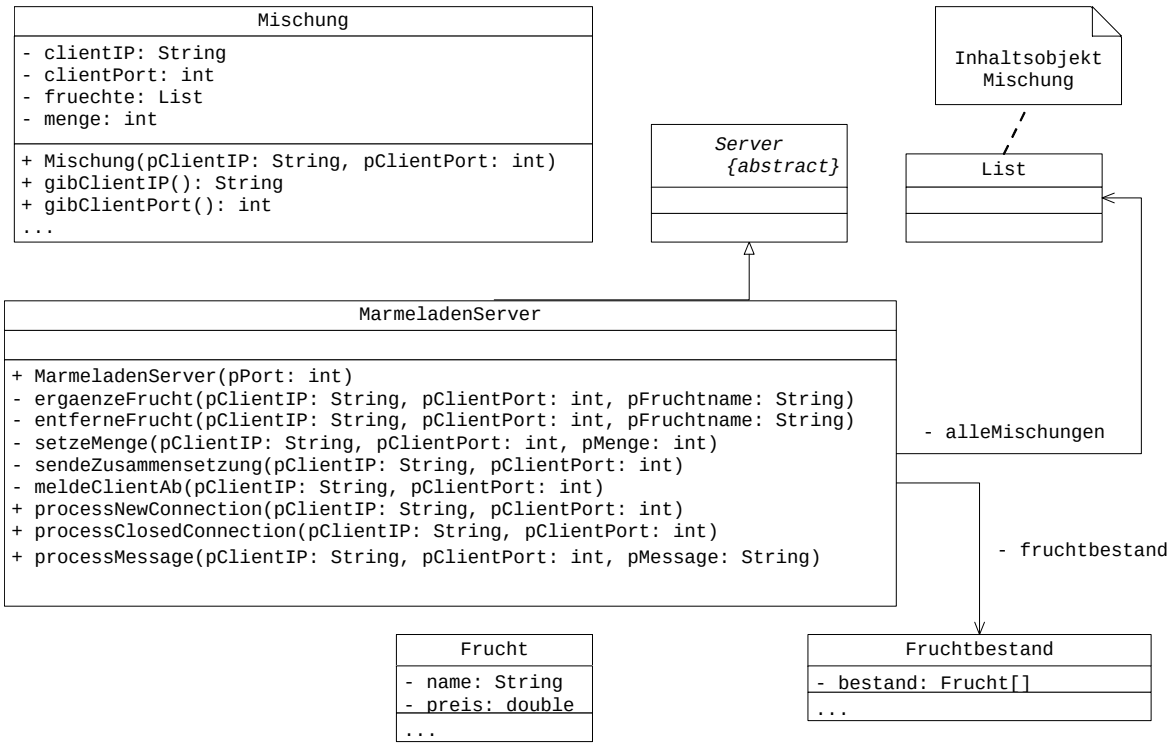


Abbildung 2: Auszug des Implementationsdiagramms

Name: _____

b) *Beschreiben Sie die Beziehungen zwischen den in Abbildung 2 dargestellten Klassen.*

Erläutern Sie zwei Gründe, dass der Server die Mischungen aller Clients in Form einer Liste verwaltet.

Modellieren Sie das vollständige Implementationsdiagramm der Klasse `Frucht` *und begründen Sie die Wahl Ihrer Methoden.*

(15 Punkte)

c) Gegeben sei die folgende Methode `wasTueIch` der Klasse `MarmeladenServer`:

```
 1 private Mischung wasTueIch(String pClientIP, int pClientPort) {
 2   Mischung ergebnisMischung = null;
 3   Mischung aktMischung;
 4   alleMischungen.toFirst();
 5   while (alleMischungen.hasAccess()
                                    && ergebnisMischung == null) {
 6     aktMischung = (Mischung) alleMischungen.getObject();
 7     if (aktMischung.gibClientIP().equals(pClientIP)
                 && aktMischung.gibClientPort() == pClientPort) {
 8       ergebnisMischung = aktMischung;
 9     }
10     alleMischungen.next();
11   }
12   return ergebnisMischung;
13 }
```

Name: _____

Der Server verwaltet in der Liste `alleMischungen` momentan die Marmeladenmischungen von vier Clients mit den folgenden Daten:

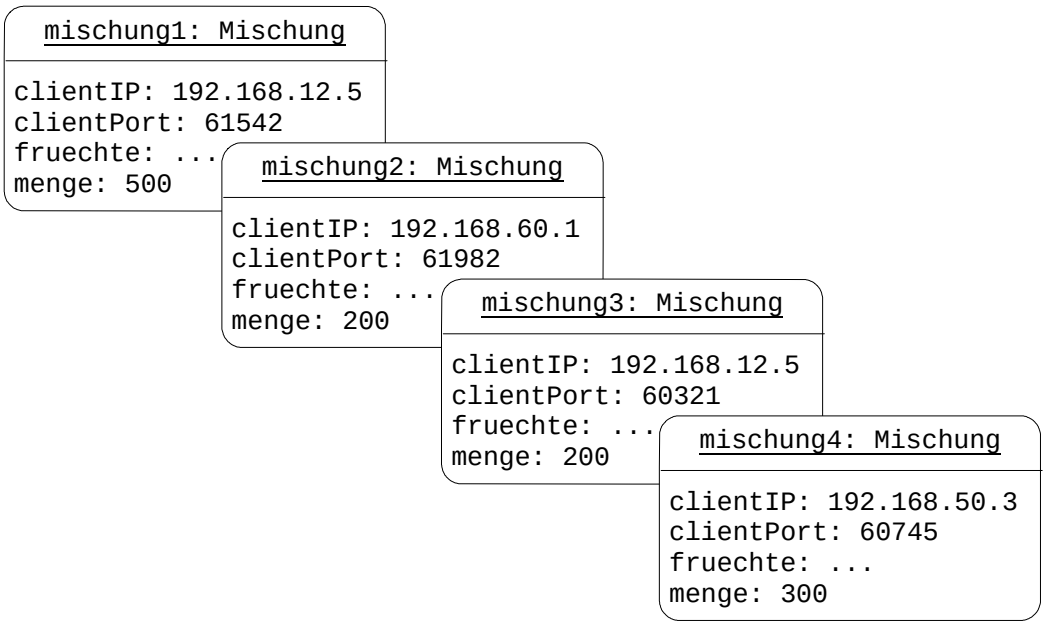


Abbildung 3: Marmeladenmischungen des Servers

Analysieren Sie die Methode `wasTueIch`*, indem Sie erklären, wie die Methode bei einem Aufruf mit den Parametern* `pClientIP = "192.168.12.5"` *und* `pClientPort = 60321` *arbeitet.*

Erläutern Sie die Funktionalität der Methode.

(8 Punkte)

d) *Implementieren Sie die Methode* `processNewConnection` *der Klasse* `MarmeladenServer`*, sodass dem Client entsprechend dem Protokoll aus Abbildung 1 die Erstanmeldung bestätigt wird und die Netzwerkeigenschaften des Clients dem Server in der Liste* `alleMischungen` *bekannt ist.*

Erläutern Sie, welche Operationen in der Methode `processClosedConnection` *erforderlich werden, damit der Marmeladenserver korrekt arbeitet.*

(10 Punkte)

Name: _____

e) Der Bioladenbesitzer stellt fest, dass immer wieder Marmeladen bestellt werden, welche dann aber nicht abgeholt werden. Er möchte deshalb, dass sich registrierte Kunden mit einem maximal 20 Buchstaben umfassenden Benutzernamen anmelden, bevor sie eine Bestellung vornehmen können. Die Übertragung des Benutzernamens soll mit dem Vigenère-Verfahren verschlüsselt erfolgen.

Der Bioladenbesitzer vereinbart dafür vorher mit jedem Kunden einen zufälligen Schlüssel bestehend aus 20 Buchstaben, mit dessen Hilfe der Benutzername verschlüsselt wird.

Stellen Sie die Verschlüsselung des Benutzernamens `ERDBEERHEXE` *mithilfe des zufälligen Schlüssels* `OEWNHQETVRNHGCWPGAWU` *dar. Verwenden Sie das in der Anlage angegebene Vigenère-Quadrat.*

Beurteilen Sie die Sicherheit der Verschlüsselung, wenn das angegebene Verfahren angewendet wird.

(8 Punkte)

Zugelassene Hilfsmittel:

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

Name: _____

Anlage

Das Vignère-Quadrat:

		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B		B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C		C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D		D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E		E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F		F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G		G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H		H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I		I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J		J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K		K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L		L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M		M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N		N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O		O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P		P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q		Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R		R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S		S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T		T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U		U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V		V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W		W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X		X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y		Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z		Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Schlüssel: **OEWNHQETVRNHGCWPGAWU**

Name: _____

Anhang

Die Klasse `MarmeladenServer`

Diese Klasse ist Unterklasse der Klasse `Server`. Sie verwaltet die zur Wahl stehenden Fruchtsorten sowie alle Mischungen der Clients, die am Server angemeldet sind. Dabei stellt sie neben den geerbten Methoden u. a. folgende Methoden zur Verfügung:

Auszug aus der Dokumentation der Klasse `MarmeladenServer`

Konstruktor **`MarmeladenServer(int pPortNr)`**

Nach dem Aufruf dieses Konstruktors bietet ein Marmeladenserver seinen Dienst über die angegebene Portnummer an. Clients können sich nun mit dem Server verbinden.

Auftrag **`void ergaenzeFrucht(String pClientIP, int pClientPort, String pFruchtname)`**

Ist pFruchtname ein gültiger Fruchtname, so wird der Mischung des Clients mit der IP **`pClientIP`** und dem Port **`pClientPort`** die entsprechende Frucht hinzugefügt. Außerdem schickt der Server dem Client über das Netzwerk eine Rückmeldung entsprechend dem Protokoll.
Ist pFruchtname kein gültiger Fruchtname, so schickt der Server dem Client eine Fehlermeldung entsprechend dem Protokoll.

Auftrag **`void entferneFrucht(String pClientIP, int pClientPort, String pFruchtname)`**

Ist pFruchtname ein gültiger Fruchtname, so wird aus der Mischung des Clients mit der IP **`pClientIP`** und dem Port **`pClientPort`** die entsprechende Frucht, sofern vorhanden, entfernt. Außerdem schickt der Server dem Client über das Netzwerk eine Rückmeldung entsprechend dem Protokoll.
Ist **`pFruchtname`** kein gültiger Fruchtname, so schickt der Server dem Client eine Fehlermeldung entsprechend dem Protokoll.

Auftrag **`void setzeMenge(String pClientIP, int pClientPort, int pMenge)`**

Ist **`pMenge`** eine gültige Mengenangabe, so wird die Menge der Mischung des Clients mit der IP **`pClientIP`** und dem Port **`pClientPort`** entsprechend geändert. Außerdem schickt der Server dem Client über das Netzwerk eine Rückmeldung entsprechend dem Protokoll.
Ist **`pMenge`** keine gültige Mengenangabe, so schickt der Server dem Client eine Fehlermeldung entsprechend dem Protokoll.

Name: _____

Auftrag `void sendeZusammensetzung(String pClientIP, int pClientPort)`

Dem Client mit der IP **`pClientIP`** und dem Port **`pClientPort`** wird die Zusammensetzung der aktuellen Mischung entsprechend dem Protokoll über das Netzwerk geschickt.

Auftrag `void meldeClientAb(String pClientIP, int pClientPort)`

Dem Client mit der IP **`pClientIP`** und dem Port **`pClientPort`** wird die Abmeldebestätigung entsprechend dem Protokoll über das Netzwerk geschickt. Danach trennt der Server die Verbindung zum Client.

Die Klasse `Mischung`

Diese Klasse verwaltet die Netzwerkeigenschaften eines Clients, der sich an dem Server angemeldet hat, sowie die Zusammensetzung der Marmeladenmischung des Clients.
Die Klasse stellt u. a. folgende Methoden zur Verfügung:

Auszug aus der Dokumentation der Klasse `Mischung`

Konstruktor `Mischung(String pClientIP, int pClientPort)`

Ein neues Objekt der Klasse Mischung wird angelegt. Das Objekt kennt die Netzwerkeigenschaften (IP und Port) des zugehörigen Clients.

Anfrage `String gibClientIP()`

Die IP des zugehörigen Clients wird zurückgegeben.

Anfrage `int gibClientPort()`

Die Portnummer des zugehörigen Clients wird zurückgegeben.

Name: _____

Die Klasse Server

Über die Klasse **`Server`** ist es möglich, eigene Serverdienste anzubieten, so dass Clients Verbindungen gemäß dem TCP/IP-Protokoll hierzu aufbauen können. Nachrichten werden grundsätzlich zeilenweise verarbeitet, d. h., beim Senden einer Zeichenkette wird ein Zeilentrenner ergänzt und beim Empfangen wird er entfernt.

Verbindungsaufbau, Nachrichtenempfang und Verbindungsende geschehen nebenläufig. Durch Überschreiben der entsprechenden Methoden kann der Server auf diese Ereignisse reagieren.

Eine Fehlerbehandlung ist in dieser Klasse aus Gründen der Vereinfachung nicht vorgesehen.

Dokumentation der Klasse Server

Konstruktor **`Server(int pPortNr)`**

Nach dem Aufruf dieses Konstruktors bietet ein Server seinen Dienst über die angegebene Portnummer an. Clients können sich nun mit dem Server verbinden.

Auftrag **`void closeConnection(String pClientIP, int pClientPort)`**

Unter der Voraussetzung, dass eine Verbindung mit dem angegebenen Client existiert, wird diese beendet. Der Server sendet sich die Nachricht `processClosedConnection`.

Auftrag **`void processClosedConnection(String pClientIP, int pClientPort)`**

Diese Methode ohne Anweisungen wird aufgerufen, bevor der Server die Verbindung zu dem in der Parameterliste spezifizierten Client schließt. Durch das Überschreiben in Unterklassen kann auf die Schließung der Verbindung zum angegebenen Client reagiert werden.

Auftrag **`void processMessage(String pClientIP, int pClientPort, String pMessage)`**

Der Client mit der angegebenen IP und der angegebenen Portnummer hat dem Server eine Nachricht gesendet. Dieser ruft daraufhin diese Methode ohne Anweisungen auf. Durch das Überschreiben in Unterklassen kann auf diese Nachricht des angegebenen Client reagiert werden.

Name: _____

Auftrag `void processNewConnection(String pClientIP, int pClientPort)`

Der Client mit der angegebenen IP-Adresse und der angegebenen Portnummer hat eine Verbindung zum Server aufgebaut. Der Server hat daraufhin diese Methode aufgerufen, die in dieser Klasse keine Anweisungen enthält. Durch das Überschreiben in Unterklassen kann auf diesen Neuaufbau einer Verbindung von dem angegebenen Client zum Server reagiert werden.

Auftrag `void send(String pClientIP, int pClientPort, String pMessage)`

Wenn eine Verbindung zum angegebenen Client besteht, dann wird diesem Client die angegebene Nachricht – um einen Zeilentrenner erweitert – gesendet.

Auftrag `void sendToAll(String pMessage)`

Die angegebene Nachricht wird – um einen Zeilentrenner erweitert – an alle verbundenen Clients gesendet.

Auftrag `void close()`

Alle bestehenden Verbindungen werden getrennt. Der Server kann nicht mehr verwendet werden.

Name: _____

Die Klasse `List`

Objekte der Klasse **`List`** verwalten beliebig viele, linear angeordnete Objekte. Auf höchstens ein Listenobjekt, aktuelles Objekt genannt, kann jeweils zugegriffen werden. Wenn eine Liste leer ist, vollständig durchlaufen wurde oder das aktuelle Objekt am Ende der Liste gelöscht wurde, gibt es kein aktuelles Objekt. Das erste oder das letzte Objekt einer Liste können durch einen Auftrag zum aktuellen Objekt gemacht werden. Außerdem kann das dem aktuellen Objekt folgende Listenobjekt zum neuen aktuellen Objekt werden.

Das aktuelle Objekt kann gelesen, verändert oder gelöscht werden. Außerdem kann vor dem aktuellen Objekt ein Listenobjekt eingefügt oder ein Listenobjekt an das Ende der Liste angefügt werden.

Dokumentation der Klasse `List`

Konstruktor **`List()`**

Eine leere Liste wird erzeugt.

Anfrage **`boolean isEmpty()`**

Die Anfrage liefert den Wert `true`, wenn die Liste keine Objekte enthält, sonst liefert sie den Wert `false`.

Anfrage **`boolean hasAccess()`**

Die Anfrage liefert den Wert `true`, wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert `false`.

Auftrag **`void next()`**

Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausführung des Auftrags kein aktuelles Objekt, d. h., `hasAccess()` liefert den Wert `false`.

Auftrag **`void toFirst()`**

Falls die Liste nicht leer ist, wird das erste Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

Auftrag **`void toLast()`**

Falls die Liste nicht leer ist, wird das letzte Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

Name: _____

Anfrage `Object getObject()`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird das aktuelle Objekt zurückgegeben, andernfalls (`hasAccess() == false`) gibt die Anfrage den Wert `null` zurück.

Auftrag `void setObject(Object pObject)`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`) und `pObject` ungleich `null` ist, wird das aktuelle Objekt durch `pObject` ersetzt. Sonst bleibt die Liste unverändert.

Auftrag `void append(Object pObject)`

Ein neues Objekt `pObject` wird am Ende der Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Wenn die Liste leer ist, wird das Objekt `pObject` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt (`hasAccess() == false`). Falls `pObject` gleich `null` ist, bleibt die Liste unverändert.

Auftrag `void insert(Object pObject)`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird ein neues Objekt vor dem aktuellen Objekt in die Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Falls die Liste leer ist und es somit kein aktuelles Objekt gibt (`hasAccess() == false`), wird `pObject` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt. Falls es kein aktuelles Objekt gibt (`hasAccess() == false`) und die Liste nicht leer ist oder `pObject` gleich `null` ist, bleibt die Liste unverändert.

Auftrag `void concat(List pList)`

Die Liste `pList` wird an die Liste angehängt. Anschließend wird `pList` eine leere Liste. Das aktuelle Objekt bleibt unverändert. Falls `pList` `null` oder eine leere Liste ist, bleibt die Liste unverändert.

Auftrag `void remove()`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird das aktuelle Objekt gelöscht und das Objekt hinter dem gelöschten Objekt wird zum aktuellen Objekt. Wird das Objekt, das am Ende der Liste steht, gelöscht, gibt es kein aktuelles Objekt mehr (`hasAccess() == false`). Wenn die Liste leer ist oder es kein aktuelles Objekt gibt (`hasAccess() == false`), bleibt die Liste unverändert.

Unterlagen für die Lehrkraft

Abiturprüfung 2014

Informatik, Grundkurs

1. Aufgabenart

Aufgabenart	Aufgabenstellungen aus dem Bereich Client-Server-Strukturen
Syntaxvariante	Java

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

- entfällt

4. Bezüge zu den Vorgaben 2014

1. *Inhaltliche Schwerpunkte*
   Modellieren und Implementieren kontextbezogener Problemstellungen als Netzwerkanwendungen
   - Netzwerkprotokolle, TCP/IP-Referenzmodell
   - Client-Server-Anwendungen
   - Kryptografie
     - Symmetrische Verschlüsselungsverfahren (Vigenère)

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- Taschenrechner
- Wörterbuch zur deutschen Rechtschreibung

[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Modelllösung

Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Zwei mögliche Fehlersituationen treten auf, wenn

1. der Kunde beim ERGAENZE- oder ENTFERNE-Kommando eine Fruchtsorte wählt, die der Server nicht im Fruchtbestand führt,
2. der Kunde beim MENGE-Kommando eine Mengenangabe ungleich 200, 300 bzw. 500 wählt.

Das vollständige Protokoll lautet dann wie folgt:

Client sendet an Server	Server sendet an Client
stellt eine Verbindung zum Server her …	`+OK Beginnen Sie mit der Mischung. Verwenden Sie die Befehle ERGAENZE, ENTFERNE, MENGE, VORSCHAU und FERTIG.`
`ERGAENZE <Fruchtname>`	`+OK Fruchtsorte <Fruchtname> hinzugefügt` `-ERR Ungültige Fruchtsorte <Fruchtname>.`
`ENTFERNE <Fruchtname>`	`+OK Fruchtsorte <Fruchtname> gelöscht` `-ERR Ungültige Fruchtsorte <Fruchtname>.`
`MENGE <Mengenangabe>`	`+OK Menge <Mengenangabe> Gramm geändert.` `-ERR Ungültige Mengenangabe <Mengenangabe> Gramm.`

`VORSCHAU`	`+OK Zutaten:<Zutat1>:<Zutat2>:...:<ZutatN>`
	`+OK Menge:<Mengenangabe> Gramm`
	`+OK Preis:<Preis> Euro`

`FERTIG`	`+OK Sie können Ihre Wunschmarmelade ab morgen in unserem Bioladen abholen.` … und trennt die Verbindung.

Das Protokoll gehört im TCP/IP-Referenzmodell zur Anwendungsschicht.

Eine Begründung ist nicht erforderlich, könnte aber wie folgt lauten: Damit das Protokoll einwandfrei funktioniert, muss sichergestellt sein, dass die Nachrichten zwischen Client und Server transportiert werden. Das heißt, die Transportschicht ist Grundlage dieses Protokolls. Deshalb kann dieses Protokoll nur zur darüber liegenden Anwendungsschicht gehören.

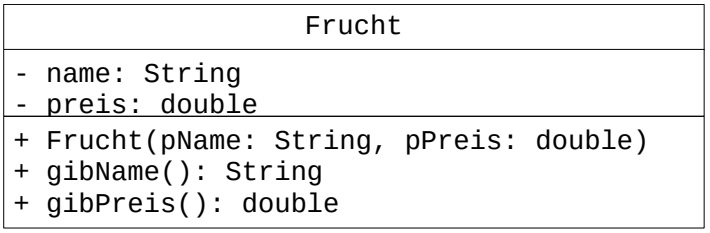
Teilaufgabe b)

Die Beziehung zwischen den Klassen `MarmeladenServer` und `Server` ist eine Vererbung, d. h., die Klasse `MarmeladenServer` ist eine Spezialisierung der Klasse `Server`.

Die Beziehungen zwischen den Klassen `MarmeladenServer` und `List` bzw. `Fruchtbestand` sind Assoziationen, d. h., Objekte der Klasse `MarmeladenServer` verwalten ein Objekt der Klasse `List` (für die Clients) sowie ein Objekt der Klasse `Fruchtbestand`.

Als Gründe für die Wahl der Liste können aufgezählt werden:
- Die Anzahl der Clients soll unbeschränkt sein. Eine Liste bietet gegenüber einem Array diesen Vorteil.
- Die Anzahl der Clients ist implizit in der Liste enthalten und muss deshalb nicht eigens verwaltet werden.
- Es wird nur der Speicherplatz reserviert, der auch wirklich benötigt wird, d. h., es bleiben keine ungenutzten Arrayfelder übrig.

Das Implementationsdiagramm der Klasse `Frucht` kann wie folgt modelliert werden:



Die Klasse muss über Methoden verfügen, die ein Setzen der Attribute `name` und `preis` ermöglichen. Dies geschieht sinnvollerweise im Konstruktor, eine Änderung der Attribute zu einem späteren Zeitpunkt scheint nicht sinnvoll. Andererseits muss es die Möglichkeit geben, die Attribute auszulesen, d. h., es werden zwei getter-Methoden für die Attribute benötigt.

Teilaufgabe c)

Zunächst wird das Funktionsergebnis der Methode auf `null` gesetzt. Danach wird die Liste aller Mischungen von Beginn an (Zeile 4) in einer Schleife (Zeilen 5 bis 11) durchlaufen. Die Schleife stoppt, wenn alle Objekte der Liste betrachtet wurden oder bereits ein Ergebnis-Objekt gefunden wurde. Im genannten Beispiel wird zunächst `mischung1` überprüft. Die IP stimmt zwar überein, der Port jedoch nicht. Dann wird `mischung2` überprüft. Hier scheitert schon der Vergleich der IPs. Dann wird `mischung3` überprüft: Sowohl IP als auch Port stimmen mit den Funktionsparametern überein, also wird das Ergebnis-Objekt `ergebnisMischung` auf das Objekt `mischung3` gesetzt. Die Schleife bricht ab, da `ergebnisMischung` ungleich `null` ist, und die Methode gibt dieses Objekt zurück.

Insgesamt wird also aus der Liste mit den Mischungen die Mischung des Clients mit der vorgegebenen IP und Port herausgesucht und zurückgegeben.

Teilaufgabe d)

```
public void processNewConnection(String pClientIP,
                                                    int pClientPort) {
  alleMischungen.append(new Mischung(pClientIP, pClientPort));
  send(pClientIP, pClientPort,
                              "+OK Beginnen Sie mit der Mischung.");
  send(pClientIP, pClientPort, "+OK Verwenden Sie die Befehle
                ERGAENZE, ENTFERNE, MENGE, VORSCHAU und FERTIG.");
}
```

Die Methode `processClosedConnection` muss sicherstellen, dass die Mischung des Clients mit der angegebenen IP und dem angegebenen Port aus der Liste `alleMischungen` gelöscht wird.

Teilaufgabe e)

Die Vigenère-Verschlüsselung lautet:

Klartext: `ERDBEERHEXE`

Schlüssel: `OEWNHQETVRNHGCWPGAWU`

Chiffrat: `SVZOLUVAZOR`

Das Verschlüsselungsverfahren ist in diesem Fall relativ sicher, da der zu verschlüsselnde Klartext kleiner ist als der verwendete Schlüssel. Zudem ist der Schlüssel zufällig erzeugt und wird nur für diese eine Aufgabe verwendet, sodass keine Häufigkeitsanalyse möglich ist.

Das Verfahren entspricht einem One-Time-Pad-Verfahren, allerdings muss dieser Begriff von dem Prüfling nicht ausdrücklich genannt werden.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	erläutert zwei Situationen für Fehlerrückmeldungen.	2			
2	gibt das vollständige Protokoll inklusive der Fehlerrück-meldungen an.	5			
3	gibt die Zuordnung zum TCP/IP-Referenzmodell an.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (9) …… ……					
	Summe Teilaufgabe a)	**9**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	beschreibt die Vererbung.	2			
2	beschreibt die Assoziationen.	3			
3	erläutert zwei Gründe für die Wahl der Liste.	4			
4	gibt das Klassendiagramm der Klasse `Frucht` an.	3			
5	begründet die Wahl der Methoden.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (15) …… ……					
	Summe Teilaufgabe b)	**15**			

[2] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert die Methode, indem er erklärt, wie die Methode arbeitet.	6			
2	erläutert die Funktionalität der gesamten Methode.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe c)	**8**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	implementiert die Methode `processNewConnection`.	6			
2	erläutert die Operationen der Methode `processClosedConnection`.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe d)	**10**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	verschlüsselt das Wort ERDBEERHEXE.	4			
2	beurteilt die Sicherheit des Verschlüsselungsverfahrens im speziellen Fall.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe e)	**8**			

	Summe insgesamt	**50**			

Festlegung der Gesamtnote (Bitte nur bei der letzten bearbeiteten Aufgabe ausfüllen.)

	Lösungsqualität			
	maximal erreichbare Punktzahl	EK	ZK	DK
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**100**			
aus der Punktsumme resultierende Note				
Note ggf. unter Absenkung um ein bis zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

ggf. arithmetisches Mittel der Punktsummen aus EK und ZK: _____

ggf. arithmetisches Mittel der Notenurteile aus EK und ZK: _____

Die Klausur wird abschließend mit der Note: _____ (____ Punkte) bewertet.

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

Für die Zuordnung der Notenstufen zu den Punktzahlen ist folgende Tabelle zu verwenden:

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	100 – 95
sehr gut	14	94 – 90
sehr gut minus	13	89 – 85
gut plus	12	84 – 80
gut	11	79 – 75
gut minus	10	74 – 70
befriedigend plus	9	69 – 65
befriedigend	8	64 – 60
befriedigend minus	7	59 – 55
ausreichend plus	6	54 – 50
ausreichend	5	49 – 45
ausreichend minus	4	44 – 39
mangelhaft plus	3	38 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0

Name: _____

Abiturprüfung 2014

Informatik, Grundkurs

Aufgabenstellung:

An einer Schule wird das Schreiben von Klausuren in der Qualifikationsphase neu organisiert. Die Schülerinnen und Schüler können zukünftig selbst entscheiden, zu welchen Zeitpunkten in einem Halbjahr sie ihre Klausuren schreiben wollen. Für jede Klausur ist festgelegt, welches Fachthema sie aufgreift, zu welchem Kurs sie gehört und in welcher Jahrgangsstufe sie geschrieben werden kann.

Die Schülerinnen und Schüler wählen dann im Laufe des Halbjahrs für ihre Kurse aus, an welchen Terminen sie die Klausuren schreiben wollen.

Um diese flexible Regelung für Klausuren verwalten zu können, wird von den Schülerinnen und Schülern der Informatikkurse eine Datenbank angelegt. Sie entwerfen das folgende Entity-Relationship-Diagramm:

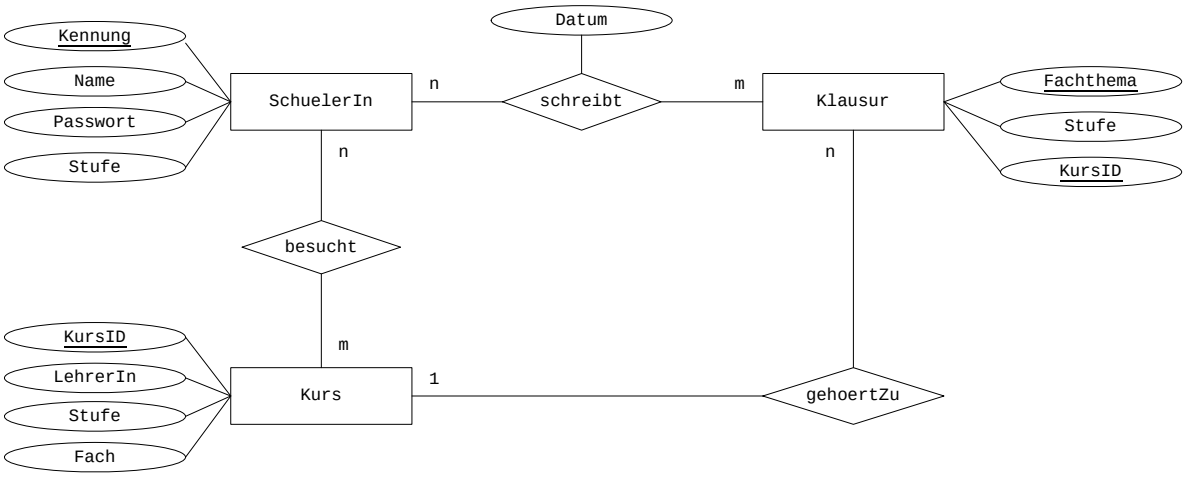


Abbildung 1: ER-Diagramm

Name: _____

Aus dem ER-Diagramm leiten die Schülerinnen und Schüler das folgende Datenbankschema ab:

SchuelerIn(Kennung, Name, Passwort, Stufe)
Kurs(KursID, LehrerIn, Stufe, Fach)
Klausur(Fachthema, ↑KursID, Stufe)
besucht(↑Kennung, ↑KursID)
schreibt(↑Kennung, ↑Fachthema, ↑KursID, Datum)

Abbildung 2: Datenbankschema

a) *Beschreiben Sie, wie die Beziehungen des Entity-Relationship-Diagramms (Abbildung 1) in dem Datenbankschema (Abbildung 2) umgesetzt wurden.*

(8 Punkte)

b) Eine kleine Arbeitsgruppe von Schülerinnen und Schülern hat bereits einige SQL-Anweisungen entwickelt. Sie haben ihre Arbeit jedoch nicht dokumentiert und nun ist unklar, welche Ergebnisse die SQL-Anweisungen liefern.

(i)

```
1 SELECT LehrerIn
2 FROM Kurs
3 WHERE Fach = "Informatik"
```

(ii)

```
1 SELECT Fach, LehrerIn, count( Kennung ) AS Anzahl
2 FROM Kurs, besucht
3 WHERE Kurs.KursID = besucht.KursID
4       AND Kurs.Fach = "Informatik"
5 GROUP BY Kurs.KursID
```

(iii)

```
1 SELECT Name, Datum
2 FROM SchuelerIn, schreibt
3 WHERE SchuelerIn.Kennung = schreibt.Kennung
4       AND Fachthema = "Lyrik"
5       AND Datum > '2014-04-30'
6 ORDER BY Datum ASC
```

Hinweis: Das Format des Datums ist wie folgt zu verstehen:
Die Reihenfolge ist `'Jahr-Monat-Tag'`. `'2014-03-30'` entspricht also dem 30. März 2014.

Name: _____

Analysieren Sie die SQL-Anweisungen, indem Sie sie auf die in der Anlage angegebenen Daten anwenden und die daraus resultierenden Tabellen angeben.

Erläutern Sie, welche Informationen mit den dargestellten SQL-Anweisungen gesucht werden.

(12 Punkte)

c) Um die Schülerinnen und Schüler gut beraten zu können, sollen unter anderem folgende Informationen aus der Datenbank verfügbar sein:

(i) Die Stufe der Schülerin `Lea Koch` soll ermittelt werden.

(ii) Die Kennungen der Schülerinnen und Schüler sollen ermittelt werden, die in den Kursen der Lehrerin `Grün` Klausuren angemeldet haben.

(iii) Es sollen die Schülerinnen und Schüler aus der Einführungsphase `EF` ermittelt werden, die noch keine Klausur angemeldet haben.

Ermitteln Sie für diese drei genannten Punkte geeignete SQL-Anweisungen, mit deren Hilfe die geforderten Informationen verfügbar werden.

(12 Punkte)

d) Das von den Schülerinnen und Schülern entwickelte Datenbankschema befindet sich nicht in der ersten Normalform – und damit auch nicht in der zweiten oder dritten Normalform. Beim Erzeugen, beim Lesen, beim Verändern und beim Löschen von Einträgen in der Datenbank können also Probleme entstehen:

Problem 1:
Es kann keine Kursliste erstellt werden, die nach Nachnamen sortiert ist. Außerdem wurde festgestellt, dass die Eingabe des Namens uneinheitlich ist:
Während meistens die Eingabe in der Form `"Vorname Nachname"` gewählt wurde, wurde auch zweimal die Form `"Nachname, Vorname"` gewählt.

Problem 2:
Der Mathematikkurs von Frau Grün findet in der Stufe `Q1` statt. Die Klausur zur Integralrechnung, die zu diesem Kurs gehört, hat aber als Stufe den Wert `EF`.

Erläutern Sie die Gründe für das Auftreten der beiden dargestellten Probleme in Bezug zu den Normalformen.

Erläutern Sie ebenfalls, wie das Auftreten dieser Probleme vermieden werden kann.

(8 Punkte)

Name: _____

e) Nach einem Probelauf sollen einige Veränderungen an der Datenbank vorgenommen werden:
   1. Unter anderem wurde ein schwerwiegendes Problem festgestellt: Die Lehrerinnen und Lehrer waren mit dem Erstellen von Klausuren sehr belastet, da meistens nur eine Person aus einem bestimmten Kurs an einem Tag Klausur geschrieben hat. Außerdem haben Schülerinnen und Schüler Termine eingetragen, an denen das Schreiben von Klausuren nicht möglich war.
      Deswegen sollen für jede Klausur eines Kurses zu einem Fachthema nun einige Termine vorgegeben werden.
   2. Von einigen Lehrerinnen und Lehrern kommt der Vorschlag, die Datenbank auch für den Eintrag der Klausurnoten zu nutzen. So könnten die Lehrerinnen und Lehrer einer Klasse immer die allgemeine Entwicklung eines Schülers oder einer Schülerin im Blick haben und andererseits könnten die Schülerinnen und Schüler sich sogar von zuhause aus über ihre Klausuren informieren.

   Entwerfen Sie für die beschriebenen Anforderungen ein Entity-Relationship-Diagramm, das das Diagramm aus Abbildung 1 erweitert und diese Veränderungen einbaut.

   Erläutern Sie Ihre Entscheidungen.

   Nehmen in Bezug auf den Datenschutz Stellung zum zweiten Veränderungsvorschlag.

(10 Punkte)

Zugelassene Hilfsmittel:

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

Name: _____

Anlage

Beispieldaten

SchuelerIn			
Kennung	**Name**	**Passwort**	**Stufe**
an.nowak	Anna Nowak	6579	EF
ju.kowalski	Julia Kowalski	7475	EF
li.fischer	Lisa Fischer	7670	EF
la.weber	Laura Weber	7687	EF
ka.meyer	Katharina Meyer	7577	EF
ka.meyer1	Katja Meyer	7577	EF
le.becker	Lena Becker	7666	EF
an.hoffmann	Annika Hoffmann	6572	Q1
ja.schäfer	Jana Schäfer	7483	Q1
le.koch	Lea Koch	7675	Q1
ma.bauer	Marie Bauer	7766	Q1
ja.wolf	Wolf, Jan	7487	Q1
ph.neumann	Philip Neumann	8078	Q1
ph.neumann1	Neumann, Philip	8079	Q1
al.zimmermann	Alexander Zimmermann	6590	Q1
da.braun	Daniel Braun	6866	Q1
to.krüger	Tobias Krüger	8475	Q1
ma.hartmann	Marcel Hartmann	7772	Q2
de.lange	Dennis Lange	6876	Q2
fl.werner	Florian Werner	7087	Q2
ni.schmitz1	Niklas Schmitz	7883	Q2
ma.krause	Maximilian Krause	7775	Q2
ha.yilmaz	Hamid Yilmaz	7289	Q2

Name: _____

Kurs			
KursID	**LehrerIn**	**Stufe**	**Fach**
1	Knüller	EF	Mathematik
2	Tink	EF	Informatik
3	Rafel	EF	Englisch
4	Heller	EF	Deutsch
5	Braun	EF	Biologie
6	Grün	Q1	Mathematik
7	Weiser	Q1	Informatik
8	Selte	Q1	Pädagogik
9	Knarowski	Q1	Französisch
10	Tingelmann	Q1	Englisch
11	Bergoglio	Q1	Relgion
12	Weizenbaum	Q1	Philosophie
13	Heula	Q1	Mathematik
14	Schmidt	Q1	Englisch
15	Schumacher	Q2	Mathematik
16	Braun	Q2	Informatik
17	Rafel	Q2	Englisch
18	Grün	Q2	Erdkunde

Klausur		
Fachthema	**Stufe**	**KursID**
Lyrik	EF	3
Lyrik	EF	4
Cytologie	EF	5
Integralrechnung	EF	6
Binärbäume	Q1	7
Lineare Strukturen	Q1	7
Netzwerke	Q2	16
Differenzialrechnung	EF	1
Globalisierung	Q1	14
Globalisierung	Q2	18

Name: _____

besucht	
Kennung	**KursID**
an.nowak	1
le.becker	1
an.nowak	2
le.becker	2
la.weber	2
an.nowak	3
le.becker	3
la.weber	3
an.nowak	4
al.zimmermann	6
da.braun	6
ph.neumann	6
ja.wolf	7
ja.wolf	10
ph.neumann	7
ja.wolf	8
ph.neumann1	10
ma.hartmann	16
ma.hartmann	17
ma.hartmann	18

schreibt			
Kennung	**Fachthema**	**KursID**	**Datum**
an.nowak	Lyrik	3	2014-04-21
an.nowak	Lyrik	4	2014-05-22
an.nowak	Cytologie	5	2014-06-13
ja.wolf	Binärbäume	7	2014-02-09
je.klein	Globalisierung	14	2014-04-09
la.weber	Lyrik	3	2014-05-21
la.weber	Monarchie	3	2014-06-21
ma.hartmann	Globalisierung	18	2014-02-13

Hinweis: Das Format des Datums ist wie folgt zu verstehen:
Die Reihenfolge ist `'Jahr-Monat-Tag'`. `'2014-03-30'` entspricht also dem 30. März 2014.

Unterlagen für die Lehrkraft

Abiturprüfung 2014

Informatik, Grundkurs

1. Aufgabenart

Aufgabenart	Aufgabenstellungen aus dem Bereich Relationale Datenbanken
Syntaxvariante	–

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

- entfällt

4. Bezüge zu den Vorgaben 2014

1. *Inhaltliche Schwerpunkte*
   Relationale Datenbanken
   - Modellieren kontextbezogener Problemstellungen als Datenbanken mit dem Entity-Relationship Modell
   - Datenbankschemata
   - Normalisierung: Überführung einer Datenbank in die 1. bis 3. Normalform
   - Relationenalgebra (Selektion, Projektion, Vereinigung, Differenz, kartesisches Produkt, Umbenennung, Join)
   - SQL-Abfragen über eine und mehrere verknüpfte Tabellen

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Modelllösungen

Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Die n:1-Beziehung `gehoertZu` ist durch den Eintrag eines Fremdschlüssels `KursID` im Relationenschema `Klausur` umgesetzt: Da jede Klausur nur einem Kurs zugeordnet ist, ist diese Vorgehensweise ausreichend.

Sowohl die Beziehung `schreibt` als auch die Beziehung `besucht` sind n:m-Relationen. Für beide wird ein Relationenschema angelegt, in dem die jeweiligen Schlüsselattribute der anderen `Relationenschemata` eintragen werden: für `besucht` also die Fremdschlüssel `Kennung` und `KursID`, für `schreibt` die Fremdschlüssel `Kennung` und `Fachthema` sowie `KursID`.

Bei der Beziehung `schreibt` wird zusätzlich im Relationenschema das Attribut `Datum` eingefügt.

Teilaufgabe b)

(i)

LehrerIn
Tink
Weiser
Braun

Es werden alle Lehrerinnen und Lehrer erfasst, die Informatik unterrichten.

(ii)

Fach	LehrerIn	Anzahl
Informatik	Tink	3
Informatik	Weiser	2
Informatik	Braun	1

Es werden die Fächer und die Namen der Lehrerinnen und Lehrer aller Informatik-Kurse ausgegeben. Außerdem wird in jeder Zeile die Anzahl der Schülerinnen und Schüler angegeben, die den jeweiligen Kurs besuchen.

(iii)

Name	Datum
Laura Weber	2014-05-21
Anna Nowak	2014-05-22

Es werden alle Schülerinnen und Schüler ausgegeben, die nach dem 30.4.2014 zum Fachthema `Lyrik` eine Klausur angemeldet haben. Das Datum der angemeldeten Klausur wird ebenfalls angegeben. Die Daten werden nach Datum aufsteigend sortiert. Die Schülerin `Anna Nowak` schreibt in zwei unterschiedlichen Fächern zum Fachthema `Lyrik`.

Teilaufgabe c)

(i)
```
SELECT Stufe
FROM SchuelerIn
WHERE Name = "Lea Koch"
```

(ii)
```
SELECT Kennung
FROM schreibt
JOIN Kurs
ON schreibt.KursID = Kurs.KursID
WHERE Kurs.LehrerIn = "Grün"
```

Alternative ohne `JOIN`:
```
SELECT Kennung
FROM schreibt, Kurs
WHERE schreibt.KursID = Kurs.KursID
      AND Kurs.LehrerIn = "Grün"
```

(iii)
```
SELECT Name, Stufe
FROM SchuelerIn
WHERE Stufe = "EF"
      AND Kennung NOT IN (
          SELECT DISTINCT Kennung
          FROM schreibt
          )
```

Teilaufgabe d)

Zum ersten Problem:

Dieses Problem tritt auf, da das Datenbankschema nicht in der ersten Normalform ist: Die Struktur der Eingabe von Vorname und Nachname scheint nicht eindeutig zu sein. Außerdem soll eine Kursliste nach Nachnamen sortiert werden. Damit ist das Attribut `Name` nicht atomar. Einen Hinweis hierauf geben die verwendeten Zeichen Komma und Leerzeichen als Trenner.

Dieses Problem kann vermieden werden, indem das zusammengesetzte Attribut `Name` in die zwei Attribute `Nachname` und `Vorname` aufgeteilt wird.

Zum zweiten Problem:

Dieses Problem tritt auf, da das Datenbankschema – wenn das erste Problem behoben ist – nicht in zweiter Normalform ist: Das Attribut `Stufe` in der Relation `Klausur` hängt nicht vom gesamten Schlüssel ab, sondern nur von dem Teilschlüssel `KursID`. Wird das Attribut bei `KursID` geändert, z. B. wenn ein neues Schuljahr anfängt, so wird es nicht automatisch auch bei `Klausur` geändert.

Das Problem kann vermieden werden, indem das Attribut `Stufe` aus der Relation `Klausur` entfernt wird.

Teilaufgabe e)

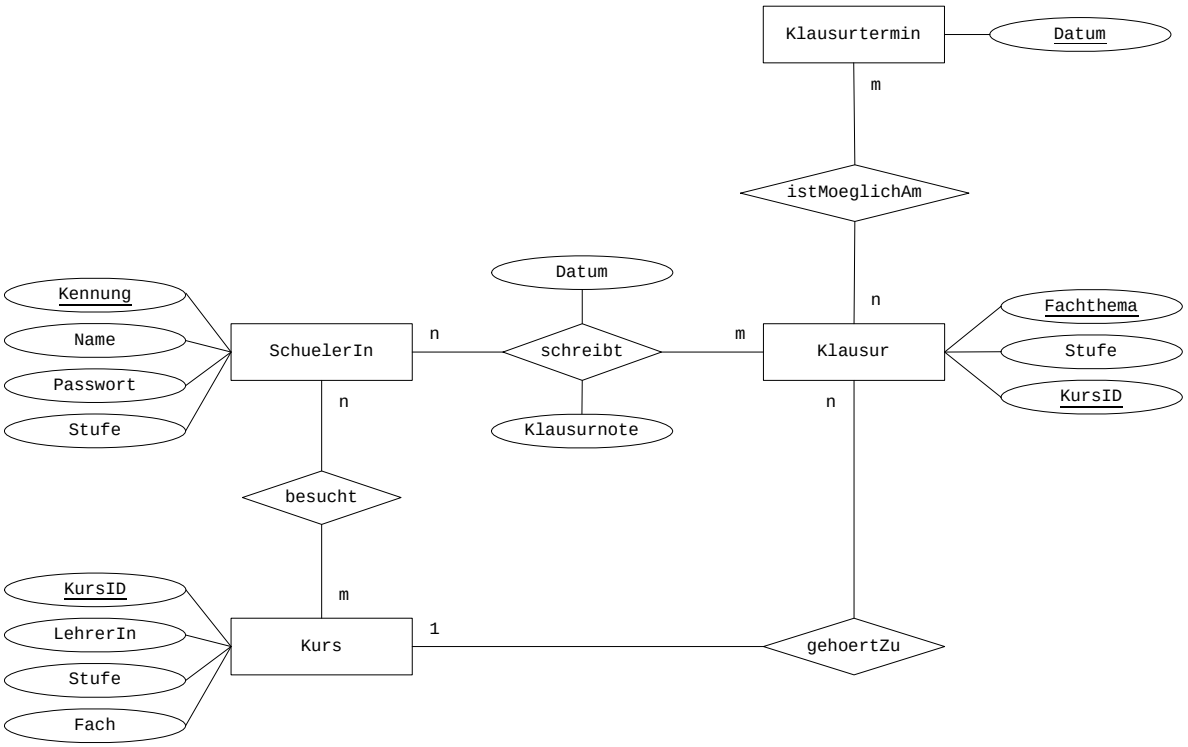


Abbildung 1: ER-Diagramm zu Aufgabenteil e)

(Die Darstellung der Erweiterung des Diagramms ist als Lösung hinreichend.)

Da jede einzelne `Klausur` in der Beziehung `schreibt` erfasst wird, ist die Erweiterung um das Attribut `Klausurnote` einfach zu bewerkstelligen: Die individuelle Note zu einer Klausur gehört direkt zu einer einzelnen Klausur und ist damit ein Attribut der Beziehung `schreibt`.

Das andere beschriebene Problem der Terminfindung lässt sich umgehen, indem für jede Klausur aus einem Pool von möglichen Terminen – dargestellt durch den Entitätstyp `Klausurtermin`, der nur das eine Attribut `Datum` enthält – beliebig viele Klausurtermine für die Schülerinnen und Schüler bereitgestellt werden. Diese Beziehung `istMoeglichAm` ist eine n:m-Beziehung, da eine Klausur eines Kurses zu einem bestimmten Fachthema sowohl an mehreren Terminen stattfinden kann als auch an einem Termin mehrere Klausuren geschrieben werden können.

Mit dieser Lösung können die Lehrerinnen und Lehrer dann auch bestimmen, an wie vielen Terminen eine Klausur in ihren Kursen möglich ist. Beispielsweise kann die Lehrkraft vier Aufgabenstellungen zu einem Thema entwickeln und für die Schülerinnen und Schüler damit vier Termine zur Wahl stellen.

Die Verarbeitung personenbezogener Daten im Rahmen von Schule – also durch eine öffentliche Verwaltung – unterliegt zwei wesentlichen Grundsätzen: Die Verarbeitung muss sich auf den erforderlichen Umfang beschränken und die Daten dürfen grundsätzlich nur für die Zwecke genutzt werden, für die sie erhoben wurden.

Der Vorschlag, die Klausurdaten in dieser Datenbank zu sammeln, ist also aus zwei Gründen kritisch zu sehen:

Besonders fraglich ist, ob das vage formulierte pädagogische Ziel der „Beurteilung der allgemeinen Entwicklung“ rechtfertigt, dass alle Klausurdaten der Schülerinnen und Schüler von allen sie unterrichtenden Lehrerinnen und Lehrern eingesehen werden dürfen. Die Verhältnismäßigkeit ist hier nicht gegeben. Zumal ist fraglich, ob (nur) aus den Klausurergebnissen eine allgemeine Entwicklung herausgelesen werden kann.

Andererseits sind die Anforderungen des Schutzes der sensiblen Daten wie Klausurnoten deutlich höher als für Klausurtermine. Hier muss einerseits eine Zugriffssicherheit gewährleistet werden: Nur der richtige Personenkreis darf Zugriff auf die Daten haben.

Andererseits muss eine Ausfallsicherheit für das System gewährleistet werden. Der Verlust der eingetragenen Daten wegen eines technischen Defekts wäre hochproblematisch.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**[2]	**ZK**	**DK**
1	beschreibt die Umsetzung der n:m-Relationen.	5			
2	beschreibt die Umsetzung der n:1-Relation.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe a)	**8**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	gibt die resultierenden Tabellen an.	6			
2	erläutert die gesuchten Informationen.	6			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe b)	**12**			

[2] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	ermittelt zu (i) geeignete SQL-Anweisungen.	4			
2	ermittelt zu (ii) geeignete SQL-Anweisungen.	4			
3	ermittelt zu (iii) geeignete SQL-Anweisungen.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe c)	**12**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert die Gründe für das Auftreten der beiden dargestellten Probleme.	4			
2	erläutert, wie die Probleme vermieden werden können.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe d)	**8**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwirft ein Entity-Relationship-Diagramm.	3			
2	erläutert die Entscheidungen.	3			
3	nimmt Stellung zum zweiten Änderungsvorschlag.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe e)	**10**			

	Summe insgesamt	**50**			

Festlegung der Gesamtnote (Bitte nur bei der letzten bearbeiteten Aufgabe ausfüllen.)

	Lösungsqualität			
	maximal erreichbare Punktzahl	EK	ZK	DK
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**100**			
aus der Punktsumme resultierende Note				
Note ggf. unter Absenkung um ein bis zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

ggf. arithmetisches Mittel der Punktsummen aus EK und ZK: _____

ggf. arithmetisches Mittel der Notenurteile aus EK und ZK: _____

Die Klausur wird abschließend mit der Note: _____ (____ Punkte) bewertet.

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

Für die Zuordnung der Notenstufen zu den Punktzahlen ist folgende Tabelle zu verwenden:

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	100 – 95
sehr gut	14	94 – 90
sehr gut minus	13	89 – 85
gut plus	12	84 – 80
gut	11	79 – 75
gut minus	10	74 – 70
befriedigend plus	9	69 – 65
befriedigend	8	64 – 60
befriedigend minus	7	59 – 55
ausreichend plus	6	54 – 50
ausreichend	5	49 – 45
ausreichend minus	4	44 – 39
mangelhaft plus	3	38 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0